Instituto Superior de Economia e Gestão
Universidade Técnica de Lisboa

MÉTODOS HEURÍSTICOS PARA O PROBLEMA DE ESCALONAMENTO DE TAREFAS NUMA MÁQUINA COM CUSTOS DE ANTECIPAÇÃO E ATRASO

Mário José Gomes de Freitas Centeno

Junho de 1993

Instituto Superior de Economia e Gestão
Universidade Técnica de Lisboa

MÉTODOS HEURÍSTICOS PARA O PROBLEMA DE ESCALONAMENTO DE TAREFAS NUMA MÁQUINA COM CUSTO DE ANTECIPAÇÃO E ATRASO

Mário José Gomes de Freitas Centeno

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre em Matemática Aplicada à Economia e Gestão

Junho de 1993

Aos meus pais.

Resumo

Nesta dissertação é estudado um problema de escalonamento de tarefas numa máquina, em que se pretendem minimizar os custos de escalonamento, que são função dos desvios entre os momentos de conclusão de cada uma das tarefas e as respectivas datas desejadas de conclusão (*due dates*).

A maior parte dos problemas de escalonamento não tem resolução em tempo polinomial conhecida. O problema estudado não só pertence à classe dos problemas *NP-Difíceis* como parece estar entre os mais difíceis dessa classe.

Uma das motivações que tem levado ao crescente interesse sobre problemas em que se consideram os custos de antecipação e atraso das tarefas (custos bilaterais) é a sua adequabilidade à modelização dos métodos de gestão industrial tipo *Just-in-Time*.

É feita uma revisão da literatura existente sobre problemas de escalonamento numa só máquina com função objectivo bilateral, destacando-se por um lado, algumas propriedades das soluções óptimas para algumas versões do problema e por outro, as suas versões para as quais existe solução em tempo polinomial conhecida.

Dada a dificuldade de resolução do problema no caso geral, a resolução de instâncias, mesmo de dimensão moderada, tem de ser feita aproximadamente, utilizando métodos heurísticos. Neste trabalho foram desenvolvidos dois algoritmos construtivos cujo desempenho foi comparado com outros, do mesmo tipo, descritos na literatura. Na tentativa de melhorar os resultados obtidos foram também implementados dois algoritmos do tipo melhorativo, um determinístico (pesquisa local) e outro aleatorizado (*simulated annealing*). Em alternativa foram ainda desenvolvidas versões aleatorizadas dos algoritmos do tipo construtivo. Com estas

versões foi possível melhorar significativamente os resultados obtidos pelos métodos construtivos sem que os tempos de cálculo crescessem de forma inaceitável.

Na parte final da dissertação é apresentado o estudo computacional de teste dos métodos implementados e que foram aplicados à resolução de instâncias do problema de diferentes dimensões e características, geradas aleatoriamente.

PALAVRAS-CHAVE: Escalonamento de Tarefas; Heurísticas; Optimização Combinatória.

Agradecimentos

Antes de mais um agradecimento muito especial à Professora Doutora Maria Teresa Chaves de Almeida pela orientação encorajadora e sempre eficaz que prestou ao longo dos meses pelos quais este trabalho se prolongou.

Às Margaridas, Morgado e Carrapa, pela leitura e sugestões sempre pertinentes que fizeram das páginas que foram instadas a comentar ao longo deste longo ano de trabalho.

À minha mãe e aos meus irmãos por tudo o que suportaram e pelo incentivar contínuo.

Ao CISEP, ao CEMAPRE e à ASIATEC pela disponibilidade na utilização dos computadores e de outro material que me foi permitida ao longo do trabalho.

Aos meus colegas de Investigação Operacional I pelas tarefas que deixei de cumprir, mas que não ficaram por fazer.

Ao Departamento de Matemática pelo excelente ambiente de trabalho proporcionado.

Finalmente à JNICT pelo suporte financeiro prestado ao longo dos dois anos de realização do mestrado e em que fui bolseiro do Programa Ciência.

Como é evidente todos os erros e omissões que nesta dissertação ainda possam constar são da exclusiva responsabilidade do seu autor.

1 - Introdução

O problema de optimização abordado neste trabalho consiste na determinação do escalonamento óptimo de um conjunto de tarefas numa máquina.

Informalmente o problema pode ser descrito da seguinte forma: tem-se um conjunto de tarefas para escalonar numa só máquina, para as quais são conhecidos o tempo de processamento e a data desejada (*due date*) para a sua conclusão. Ao não ser cumprida exactamente essa *due date* incorre-se numa penalização.

Sabendo-se que a máquina apenas suporta o processamento de uma tarefa de cada vez, que não pode parar desde o momento em que se inicia o processamento da primeira tarefa até ao momento de conclusão da última tarefa e que as tarefas não podem ser interrompidas, o objectivo é determinar a sequência de tarefas que minimiza a soma das penalizações sofridas.

O problema com estas características é designado abreviadamente por *MCTDM* (Minimização dos Custos Totais de Desvio numa Máquina).

Começa-se por fazer o enquadramento do problema de escalonamento de tarefas numa máquina, com custos lineares de antecipação e atraso e *due dates* conhecidas, dentro da classe dos problemas de escalonamento de tarefas. São apresentadas duas formalizações para o *MCTDM*.

Descrevem-se as principais características do *MCTDM*, algumas propriedades de casos particulares, utilizáveis no desenvolvimento de regras de resolução heurística e apresentam-se alguns problemas de optimização com penalidades bilaterais para os quais é conhecida solução em tempo polinomial.

É feita uma breve referência aos métodos de resolução por enumeração implícita e são referenciadas as regras utilizadas na sua resolução por algoritmos de pesquisa em árvore.

No final do Capítulo 2 é apresentada uma condição de adjacência das tarefas numa sequência que é utilizada também no desenvolvimento das regras heurísticas de resolução.

Em seguida são descritos os métodos heurísticos propostos para resolução do problema.

São apresentados alguns algoritmos heurísticos de tipo construtivo e de tipo melhorativo, de que são testadas versões determinísticas e aleatorizadas.

Nos algoritmos construtivos, é feita a descrição e implementação de algumas regras já existentes na literatura e são propostos alguns novos procedimentos.

São aplicados aos algoritmos construtivos algumas técnicas de aleatorização controlada do processo de construção da solução final, com aplicação conhecida a outros problemas de optimização combinatória, mas de que não se conhecem aplicações ao *MCTDM*.

Quanto aos algoritmos melhorativos, são também ensaiadas duas alternativas: uma determinística (pesquisa local) e uma aleatorizada (*simulated annealing*).

A parte final da dissertação é dedicada à apresentação da experiência obtida com a implementação computacional dos vários algoritmos num computador do tipo IBM-Compatível. Todos os algoritmos foram aplicados a um conjunto de instâncias-teste geradas aleatoriamente de acordo com as regras usadas por outros autores que estudaram o mesmo tipo de problema de escalonamento.

2 - Problemas de Escalonamento de Tarefas

Os problemas de escalonamento de tarefas e os métodos informais para a sua resolução são tão antigos quanto a actividade humana e decorrem da necessidade de afectação de recursos ao longo do tempo de forma a ser processado um conjunto de actividades.

Na literatura de Investigação Operacional, apenas a partir de meados da década de cinquenta o seu tratamento assumiu um carácter sistemático e formal.

Uma definição genérica dos problemas de escalonamento de tarefas pode ser a seguinte: num dado sistema de processamento pretende-se efectuar o escalonamento de um conjunto de tarefas, respeitando determinadas condições, de modo a encontrar uma sequência que optimize o valor de uma dada função objectivo.

No quadro da definição apresentada, os problemas de escalonamento podem ser classificados de várias formas: *estáticos / dinâmicos*, *determinísticos / estocásticos*, *um só produto / múltiplos produtos*, *um só processador / múltiplos processadores* e *uma só operação por tarefa / múltiplas operações* (Cheng e Gupta, 1989).

Um determinado problema diz-se estático, em oposição a um problema dinâmico, se o conjunto de tarefas a processar for conhecido no momento em que se inicia o processamento.

Se a informação que define uma dada instância do problema é completamente conhecida no momento de início do processamento, então este diz-se determinístico.

O número de operações em máquinas distintas que é necessário processar para cada tarefa também permite definir diferentes

classes de problemas de escalonamento. Caso as tarefas sejam compostas por um conjunto de operações podem surgir três situações: se a ordem pela qual as operações se realizam for arbitrária para todas as tarefas o problema designa-se de *Open Shop*, se essa ordem for idêntica para todas as tarefas trata-se de um problema de *Flow Shop* e se a ordem das operações puder diferir de tarefa para tarefa diz-se um problema de *Job Shop*.

Atendendo a que, para cada uma das configurações do sistema de processamento e das tarefas referenciadas, se podem optimizar diferentes objectivos, o conjunto de problemas existente é tão vasto que a sua simples enumeração se torna difícil de concretizar.

No universo dos problemas de escalonamento a classe dos problemas com uma só máquina constitui uma classe de grande importância à qual tem sido dedicado muito trabalho de investigação.

Dentro desta classe de problemas é possível classificar as diferentes abordagens realizadas relativamente à função objectivo que se pretende optimizar ou às restrições impostas ao escalonamento.

De acordo com Gupta e Kyparisis (1987), as classes de problemas numa só máquina que têm merecido um maior destaque na literatura são aquelas que, envolvendo ou não as *due dates*, consideram a optimização de uma determinada função dos momentos de conclusão das tarefas.

No primeiro grupo estão incluídos objectivos como a minimização do número de tarefas atrasadas, do atraso máximo, dos custos totais de antecipação e atraso ou ainda a optimização da afectação das *due dates* às tarefas.

Quando a avaliação de uma dada sequência é obtida sem utilização das *due dates* estamos perante problemas em que se minimiza uma função, linear ou não, dos momentos de conclusão das tarefas.

Vários trabalhos de síntese têm sido realizados nos últimos anos, sendo de destacar os mais recentes de Baker e Scudder (1990), Sen e Gupta (1984), Fry et al (1989) e Gupta e Kyparizis (1987). Se, por um lado, é demonstrado o crescente interesse sobre este problema, medido no número de artigos publicados, por outro revela-se uma quase total ausência de esforço de unificação teórica das formas de abordagem do problema.

2.1 - Problema de Minimização dos Custos Totais de Desvio numa Máquina (*MCTDM*)

De entre os problemas de escalonamento de tarefas definidos anteriormente de uma forma genérica, considera-se neste trabalho o de Minimização dos Custos Totais de Desvio numa Máquina (*MCTDM*): dado um conjunto de tarefas para escalonar numa só máquina (processador), pretende-se determinar uma ordem de processamento que verifique um conjunto de restrições, de modo a minimizar a soma das penalidades associadas ao não cumprimento da *due date* (data pretendida para término) de cada tarefa.

Considere-se:

$N = \{1, 2, \ldots, n\}$, conjunto das tarefas a escalonar;

n, número de tarefas a escalonar;

e os seguintes parâmetros que definem cada tarefa j:

p_j, tempo de processamento;

d_j, *due date* (data pretendida para término);

a_j, penalidade de antecipação por unidade de tempo;

b_j, penalidade de atraso por unidade de tempo;

sendo p_j, d_j, a_j, b_j números inteiros não negativos.

As restrições que a ordem de processamento das tarefas deve respeitar são:

R1 - em cada momento a máquina admite apenas uma tarefa em processamento;

R2 - não é possível iniciar o processamento de uma tarefa antes do momento zero nem terminá-lo depois do momento $M = \sum_{j \in N} p_j$.

O efeito que cada grupo de restrições tem sobre os resultados do escalonamento não é idêntico.

Enquanto que algumas destas restrições aderem bem à realidade, no sentido em que não as incluir significaria considerar problemas com características essencialmente teóricas, outras correspondem a considerar apenas parte das situações que se podem verificar na realidade.

A impossibilidade de ter mais do que uma tarefa em processamento é uma restrição de carácter prático, que se justifica pela própria definição de "máquina". Admitir a hipótese de processar mais do que uma tarefa em simultâneo na mesma máquina afastar-se-ía bastante mais da realidade.

Da mesma forma, as restrições R3 também não são de difícil aceitação, já que a impossibilidade de interrupção do processamento de uma tarefa pode ser assumida como uma questão prática, se suposermos que cada tarefa é a operação mais atomística que a máquina pode processar.

São as restrições que impedem a inserção de tempo de paragem na máquina aquelas que podem constituir o mais importante afastamento da realidade. Se nalguns sistemas os custos de arranque são de tal modo elevados que é possível excluir *a priori* soluções com tempos de inactividade entre o processamento das diversas tarefas, noutros sistemas esses custos não têm expressão que justifique tal exclusão *a priori*. Estas restrições influenciam os resultados do escalonamento não apenas ao nível do valor da solução óptima como também da própria permutação (ver Davis e Kanet (b)).

Com a imposição deste tipo de restrições e utilizando a classificação de Kahlbacher (1993), o problema diz-se restrito.

Na notação mais utilizada para problemas de escalonamento, apresentada em Lawler et al (1989), o *MCTDM* é representado por

n/1/ /ET (n tarefas / uma máquina / não é permitida a interrupção do processamento das tarefas, não há relações de precedência, todas as tarefas estão prontas para escalonamento no momento zero e os tempos de processamento são inteiros não negativos / os custos considerados são os de antecipação e atraso).

É possível apresentar uma outra formalização para o problema com recurso a variáveis binárias, tal como é feito em Sousa e Wolsey (1992).

A principal motivação desta formalização é a sua semelhança com a do problema de selecção inter-temporal de projectos sujeitos a restrições de recursos.

Esta formalização põe em evidência o duplo sentido prático que este problema pode ter, seja na área do planeamento da produção seja na área da selecção de projectos.

Considerando o horizonte de planemento constituído por M períodos unitários de tempo a formalização do *MCTDM* pode ser:

$$\min \sum_{j=1}^{n} \sum_{t=1}^{M-p_j+1} [\, a_j * (d_j-p_j-t)^+ * x_{jt} + b_j * (t+p_j-d_j)^+ * x_{jt} \,] \quad (1)$$

sujeito a,

$$\sum_{t=1}^{M-p_j+1} x_{jt} = 1, \quad \forall\, j \in N \quad (2)$$

$$\sum_{j=1}^{n} \sum_{t-p_j < s \le t} x_{js} \le 1, \quad \forall\, t \in \{1,2,\ldots,M\} \quad (3)$$

$$x_{jt} \in \{0,1\}, \quad \forall j,t \quad (4)$$

onde $x_{jt}=1$ se a tarefa j tem o seu início no período t e 0 caso contrário. Os parâmetros a_j, b_j, d_j e p_j, são definidos como anteriormente.

A função objectivo representa a minimização das penalidades de desvio $(t+p_j-d_j)^+ = max\ \{t+p_j-d_j,\ 0\} = T_j$ e $(d_j-p_j-t)^+ = max\ \{d_j-p_j-t,\ 0\} = E_j$.

O primeiro grupo de n restrições (2) garante que todas as tarefas são processadas dentro do horizonte temporal M para o qual se está a fazer o escalonamento.

As M restrições (3) garantem que há apenas uma tarefa em processamento de cada vez e que nenhuma tarefa é interrompida, isto é, depois de começada uma tarefa só se dá início a outra depois de aquela estar concluída.

2.2 - Motivações para o Estudo deste Problema

Na literatura dos problemas de escalonamento de tarefas são mais tradicionais as abordagens em que não se incluem os custos de antecipação nos objectivos a minimizar, pelo que a função objectivo é não decrescente com o momento de conclusão das tarefas.

Essas abordagens são adequadas a uma grande variedade de problemas do mundo real. No entanto, existe um conjunto de situações de importância hoje em dia crescente, em que os custos de conclusão antecipada de uma tarefa são comparáveis aos do seu atraso, pelo que dificilmente se justificaria a sua não inclusão nos problemas estudados.

Os custos de antecipação estão ligados essencialmente a custos financeiros suportados com investimentos realizados antecipadamente.

Se os resultados de um dado problema de escalonamento forem, por exemplo, cash-flows actualizados gerados com uma sequência de projectos, o modelo aplicado deverá contemplar o peso que as taxas de actualização têm sobre esses mesmos resultados e que tenderá a penalizar os projectos em que sejam realizados investimentos antecipados.

Uma outra situação em que os custos de antecipação podem ser significativos, mas que é específica de certos processos produtivos, especialmente na indústria química, prende-se com a possiblidade de deterioração dos produtos resultantes de uma dada actividade cuja conclusão antecipada não é desejável.

Uma outra aplicação deste tipo de modelos é aquela que é encontrada nos recentes modelos de gestão baseados na ideia do *Just-in-Time* (*JIT*).

O *JIT* é definido como um método de eliminação das actividades não geradoras de valor acrescentado, com o objectivo de racionalizar os investimentos improdutivos realizados na actividade das empresas (Heiko, 1989 e Karmarkar, 1989).

A mobilização de recursos utilizados na constituição de stocks nos diferentes estágios da produção (matérias-primas, produtos-em-curso e produtos acabados), é uma das actividades mais improdutivas nas empresas, pelo que muitas vezes se confunde o *JIT* com os modelos de "stocks nulos", quando esta característica constitui apenas uma das componentes do *JIT*.

O objectivo de minimização dos custos financeiros com actividades improdutivas é conseguido por um lado, através da definição rigorosa dos momentos de início e conclusão de cada actividade e por outro lado, com alterações qualitativas na organização da empresa.

Assim sendo, o *JIT* compreende um conjunto de princípios mais vasto do que aqueles que se possam considerar modelizados no *MCTDM*, mas aparentemente este problema consegue captar aquilo que o *JIT* tem de mais essencial.

Os custos de atraso são mais facilmente identificáveis e englobam não só custos financeiros mas também custos comerciais. Estão ligados quer a situações de perda de vendas e de atraso em recebimentos, quer a situações de mais difícil quantificação como as de perda de imagem junto dos clientes.

2.3 - Características e Casos Particulares do Problema MCTDM

Nesta secção apresentam-se os problemas de escalonamento mais relevantes de entre os que tomam em consideração as penalidades de antecipação e atraso, designados por problemas com penalidades bilaterais.

Dada a dificuldade de resolução do *MCTDM* e o desconhecimento de propriedades das suas soluções óptimas, são analisados nesta secção alguns casos particulares que, ou têm resolução em tempo polinomial conhecida, ou sendo *NP-Difíceis* existem propriedades conhecidas para as respectivas soluções óptimas.

Descrevem-se algumas propriedades comuns a problemas com penalidades bilaterais e que são utilizadas na condução de processos heurísticos construtivos apresentados no Capítulo 3 deste trabalho.

São ainda apresentados casos particulares de problemas com penalidades bilaterais para os quais são conhecidas formas polinomiais exactas de resolução: *due dates* consideradas variáveis endógenas e função objectivo em que se minimiza a penalidade máxima.

É feita uma referência aos métodos existentes para a optimização da inserção de tempo de paragem na máquina, dada uma permutação das tarefas, por constituir a ligação entre o *MCTDM* e um dos seus casos particulares em que são omitidas as restrições R2 (a versão não restrita do *MCTDM*).

2.3.1 - Dificuldade Computacional

O resultado mais geral sobre a complexidade computacional de problemas do tipo *n/1/ /ET* com *due dates* exógenas foi apresentado por Garey, Tarjan e Wilfong (1988).

Garey et al demonstraram que o problema de minimizar a soma não ponderada dos desvios entre o momento de conclusão e a *due date* de cada tarefa é um problema *NP-Difícil*.

Tratando-se este problema de um caso particular do *MCTDM*, este resultado é-lhe generalizável, podendo então concluir-se que também o *MCTDM* é *NP-Difícil*.

Uma demonstração alternativa é obtida pela generalização do resultado de Hall e Posner (1991). Nesse trabalho considera-se um problema de *due date* comum e que se demonstra ser *NP-Difícil*.

2.3.2 - Problemas com Penalidades Lineares Bilaterais

Na classe dos problemas com penalidades bilaterais podem ser definidos vários objectivos distintos, dependendo dos valores atribuídos aos parâmetros e das variáveis de decisão incluídas.

Considerar as *due dates* endógenas pode fazer sentido se o sistema descrito fôr o de uma negociação de vendas com o cliente em que se têm que definir *due dates* para as entregas mas em que a definição de uma *due date* muito tardia é penalizada com uma redução de preço.

Considerar uma única *due date* pode representar situações em que o conjunto de tarefas constitui uma encomenda de um único cliente.

Considerar penalidades de antecipação diferentes das penalidades de atraso põe em evidência o conteúdo de cada um dos custos específicos (investimento antecipado versus perda de cliente, por exemplo).

Nos problemas com penalidades bilaterais as penalidades por antecipação são não crescentes com os momentos de conclusão das tarefas, C_j, e as penalidades por atraso são não decrescentes com esses momentos (ver Gráfico nº1).

Todas as medidas da qualidade de uma dada sequência, em que as funções penalidades possam decrescer com os momentos de conclusão das tarefas, são não regulares, segundo a terminologia introduzida por Conway, Maxwell e Miller em 1967 (Raghavachari, 1988).

Existe um outro tipo de medidas (tais como atraso médio, atraso máximo e número de tarefas atrasadas) que ignoram o custo associado ao término de uma tarefa antes da sua *due date* e em que o custo de sequenciar uma dada tarefa é não decrescente com o seu momento de término. Essas medidas dizem-se regulares.

A característica mais difícil de tratar matematicamente em todas as funções de penalidades bilaterais é a de serem não regulares.

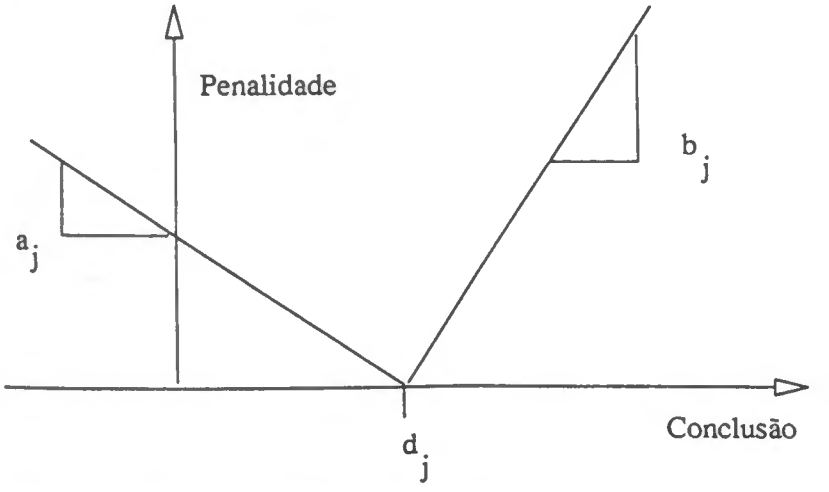


GRÁFICO Nº 1 - FUNÇÃO DE PENALIDADE BILATERAL

Este tipo de funções provoca grandes dificuldades na resolução dos problemas que a elas estão associados. *"In spite of the significance of non-regular measures of performance, very little analytical work has been done in this area. This is, in part, due to the difficulty in solving such problems. Generally, conditions of interchangeability of two jobs depend on the past scheduling pattern" - (Raghavachari, pg. 145, 1988).*

Em particular, o facto de a função objectivo ser não regular impede a generalização a este problema de alguns importantes teoremas de ordenação das tarefas, desenvolvidos por Emmons (1969), Lawler (1977) e Rinnooy Kan (1975) e que facilitariam a sua resolução por algoritmos com recurso a uma enumeração implícita das soluções admissíveis do problema.

Uma classificação possível para problemas de penalidades bilaterais lineares com due date exógena, considera dois grupos de problemas: um em que são agrupados os problemas em que todas as tarefas têm uma *due date* comum e um outro em que pelo menos uma das tarefas tem *due date* distinta das restantes.

Para o primeiro grupo de problemas são conhecidas várias propriedades das suas soluções óptimas, enquanto que para os problemas com *due dates* distintas não são conhecidas quaisquer propriedades gerais das soluções óptimas.

Para cada um destes grupos de problemas têm sido estudadas versões que consideram diferentes parâmetros para as penalidades de antecipação e atraso: problemas com penalidades idênticas (ver Hall, Kubiak e Sethi (1991) e Abdul Razaq e Potts (1988)); problemas com penalidades de atraso distintas das de antecipação mas idênticas entre as tarefas (ver Panwalkar, Smoth e Seidman (1982), Fry, Leong e Rakes (1987) e Kahlbacher (1993)); e problemas em que as penalidades de antecipação e de atraso são distintas para cada tarefa (ver Quaddus (1987), Ow e Morton (1988) e Hall e Posner (1991)).

Apresenta-se em seguida um conjunto de propriedades gerais para os problemas com *due date* comum.

2.3.2.1 - Propriedades dos Problemas de Penalidades Lineares Bilaterais com *Due Date* Comum

Para os problemas com penalidades lineares bilaterais não restritos e *due date* comum é possível definir um conjunto de propriedades das soluções óptimas, quer para os casos em que a *due date* é restritiva quer para os casos em que a *due date* é não restritiva (ver, por exemplo, Hall, Kubiak e Sethi (1991) e Hall e Posner (1991), respectivamente).

Uma *due date*, d, diz-se não restritiva sempre que permite que todas as tarefas sejam sequenciadas de forma a terminar antes de d, o que acontece sempre que $d \geq \sum_{j \in N} p_j$.

Deste conjunto de propriedades, optou-se pela apresentação de três, que têm alguma importância na definição dos métodos heurísticos de resolução apresentados no Capítulo 3.

As propriedades descritas são válidas para as versões dos problemas com *due date* comum, d, em que esta é não restritiva.

É possível demonstrar que há uma solução óptima para o problema não restrito com as seguintes propriedades (Baker e Scudder, 1990):

PROPRIEDADE I: Não há paragens na máquina, desde que é iniciado o processamento da primeira tarefa. Assim, se a tarefa i precede imediatamente a tarefa j numa dada sequência, então $C_j = C_i + p_j$.

PROPRIEDADE II: Uma das tarefas termina precisamente na *due date*.

PROPRIEDADE III: A sequência óptima tem a forma de *V* invertido, ou seja, as tarefas completadas antes da *due date* são sequenciadas por ordem não decrescente do quociente entre a penalidade por antecipação e o seu tempo de processamento e as que são completadas depois da *due date*, por ordem não crescente do quociente entre a penalidade por atraso e o seu tempo de processamento.

Decorre destas propriedades que a procura de uma sequência óptima, pode ser limitada às sequências que verifiquem a regra do V-invertido, em que uma das tarefas termine na *due date* e em que não haja tempo de paragem na máquina desde que começa o processamento da primeira tarefa.

Um importante resultado foi apresentado por Kahlbacher (1993), onde se prova a extensão das Propriedades I e III, a um problema de escalonamento em que é possível ter *due dates* distintas entre as tarefas, mas obrigando a que estas sigam um esquema do seguinte tipo: $d_j = d + p_j$ para $j = 1,2,...,n$ (modelo conhecido na literatura de escalonamento como *Equal Slack Due Date Rule*).

2.3.3 - Problemas de Resolução Polinomial

Os problemas de penalidades bilaterais que têm resolução polinomial são de dois tipos: aqueles em que as *due dates* são consideradas variáveis endógenas e os que têm como função objectivo a minimização da penalidade máxima.

É analisado ainda nesta secção o problema de inserção de tempo de paragem na máquina, dada uma permutação das tarefas, que também tem resolução em tempo polinomial conhecida.

2.3.3.1 - Problemas com *Due Date* Endógena

O problema com penalidades lineares bilaterais em que as *due dates* são variáveis de decisão, tem resolução em tempo polinomial conhecida para algumas das suas versões (ver Cheng e Gupta, 1989).

Por considerarem versões do problema semelhantes ao *MCTDM*, destacam-se os trabalhos de Seidman, Panwalkar e Smith (1981), Panwalkar, Smith e Seidman (1982), Quaddus (1987) e Chand e Chhajed (1992).

A primeira abordagem referenciada deste problema é a de Seidman et al que consideram o problema de:

$$\min Z(S, d_{[1]}, \ldots, d_{[n]}) = \sum_{j \in N} [\, a*E_{[j]} + b*T_{[j]} + c*(d_{[j]} - d_0)^+]$$

sujeito às restrições R1 e R3. O parâmetro d_0 representa o valor máximo desejável para as *due dates* e c representa a penalidade associada a uma fixação de *due dates* após d_0. O índice $[j]$ representa a tarefa sequenciada na j-ésima posição da sequência S.

Nesse trabalho é demonstrado que numa permutação óptima as tarefas são sequenciadas por ordem crescente de tempos de processamento, (regra de escalonamento habitualmente designada por *Shortest Processing Times*, *SPT*), e que a primeira tarefa é iniciada no momento zero.

O algoritmo desenvolvido é composto por dois passos: no primeiro é construída a sequência óptima e no segundo é feita a afectação óptima das *due dates* às tarefas.

O primeiro passo consiste num procedimento que constrói uma sequência segundo a ordem *SPT*. Iniciando-se no período zero o

processamento da primeira tarefa da sequência, são conhecidos os períodos de conclusão de cada tarefa.

No passo 2, as *due dates* são atribuídas a cada tarefa segundo a regra: Se $b \leq c$ então $d_{[j]} = C_{[j]}$, caso contrário $d_{[j]} = \min \{d_0, C_{[j]}\}$.

Demonstra-se que esta afectação de *due dates* distintas a cada tarefa é a que minimiza $Z(S, d_{[1]}, \ldots, d_{[n]})$.

Os mesmos autores realizaram um estudo semelhante para o problema de afectação de uma *due date* comum a todas as tarefas, Panwalkar, Smith e Seidman (1982).

O problema considerado pode ser formalizado da seguinte forma:

$$\min Z(S,d) = \sum_{j \in N} [\, a*E_{[j]} + b*T_{[j]} + c*d \,]$$

sujeito a

$$\begin{cases} d \in \mathbb{R} \\ S \in \pi \end{cases}$$

É apresentado um conjunto de resultados sobre a *due date* óptima, d^*, dos quais se destacam:

Resultado 1: Se $c \geq b$, então $d^* = 0$ e a sequência obtida pela regra *SPT* é optima.

Resultado 2: Para uma dada sequência S há uma valor óptimo para a *due date* e que é igual a $C_{[k]}$ em que k é o menor inteiro superior ou igual a $n * \frac{(b - c)}{(b + a)}$.

A partir destes resultados é desenvolvido um algoritmo em que, no primeiro passo, é determinado o número k de tarefas não atrasadas. No segundo passo são calculadas penalidades

posicionais e a partir destas a sequência óptima e o valor óptimo da *due date*, igual a $C_{[k]}$.

O trabalho de Quaddus (1987), na sequência do que havia sido feito por Cheng (1986), analisa apenas o problema da determinação da *due date* óptima através de uma formalização do problema em Programação Linear.

O problema considerado tem a seguinte formalização:

$$\min Z(d) = \sum_{j \in N} [a_{[j]} * E_{[j]} + b_{[j]} * T_{[j]} + c_{[j]} * d]$$

sujeito a

$$\begin{cases} d + T_{[j]} - E_{[j]} = C_{[j]}, & \forall\, j \in N \\ d,\ T_{[j]},\ E_{[j]} \geq 0, & \forall\, j \in N \end{cases}$$

No trabalho de Quaddus considera-se apenas uma variável de decisão, a *due date*, admitindo-se penalidades de antecipação e atraso distintas entre as tarefas.

Recorrendo à teoria da dualidade em Programação Linear, Quaddus apresenta dois Teoremas que permitem resolver optimalmente o problema formalizado.

TEOREMA Q1: Se $\sum_{j=1}^{n} c_{[j]} \geq \sum_{j=1}^{n} b_{[j]}$ então $d^* = 0$.

TEOREMA Q2: Dada uma sequência S, há um valor óptimo para a *due date* igual a $C_{[k]}$ em que k é o menor inteiro tal que

$$\sum_{j=1}^{k} [a_{[j]} + b_{[j]}] \geq \sum_{j=1}^{n} [b_{[j]} - c_{[j]}] \ .$$

Chand e Chhajed (1992), analisaram uma versão do problema em que é conhecido previamente o número de *due dates* distintas e o número de tarefas a afectar a cada *due date*. Este problema constitui uma situação intermédia entre a versão considerada por Seidman et al, uma *due date* distinta para cada tarefa, e a de Panwalkar et al, uma única *due date*.

A função objectivo a minimizar é:

$$Z(D,I,S) = \sum_{i=1}^{m} \sum_{j \in I_i} (a*E_{[j]} + b*T_{[j]} + c*D_{[j]})$$

em que I_i é o conjunto de tarefas às quais é afectada à *due date* i.

O procedimento de determinação das *due dates* óptimas e da sequência óptima constitui uma generalização do procedimento definido em Panwalkar et al.

Recentemente Panwalkar e Rajagopalan (1992), consideraram um problema em que os tempos de processamento das tarefas podem ser reduzidos de forma controlada incorrendo-se nesse caso num custo adicional, por unidade de tempo de redução. O algoritmo apresentado tem as mesmas características do de Panwalkar et al (1982) com cálculo de penalidades posicionais para as tarefas.

Não se encontraram na literatura análises da afectação de *due dates* para problemas em que os parâmetros sejam definidos com a generalidade do *MCTDM*.

2.3.3.2 - Minimização da Penalidade Máxima

O problema da minimização da penalidade máxima foi um dos primeiros problemas da classe de problemas *n/1/ /ET* a ser abordado.

O primeiro trabalho referenciado na literatura foi o de Sidney (1977), a que se seguiu o de Lakshminarayan et al (1977).

O problema tem como objectivo a minimização da seguinte função:

$$Z(S) = \max_{j \in N} [\, g(\max_{j \in N} E_j) , h(\max_{j \in N} T_j)]$$

sujeita às restrições R1 e R3.

As funções $g(E_j)$ e $h(T_j)$ representam, respectivamente, a penalidade por antecipação e a penalidade por atraso e são não decrescentes e contínuas.

Para cada tarefa é conhecido o momento desejado de início do seu processamento, e_j, para além da *due date* e do tempo de processamento. É imposta a restrição de que se para um par de tarefas i e j, $e_i < e_j$ então $d_i \leq d_j$.

A resolução do problema é baseada numa propriedade das sequências óptimas e num lema que permite o cálculo de um minorante do valor de cada sequência admissível.

A propriedade garante que, numa permutação óptima, as tarefas são sequenciadas por ordem não decrescente dos momentos desejados de início do seu processamento, o que equivale a sequenciar por ordem não decrescente de *due date* (regra de escalonamento habitualmente designada por *Earliest Due Date, EDD*).

O lema apresentado permite definir um minorante do valor de cada sequência admissível, que é dado por $g(E^*)$, onde E^* e T^* são números não negativos tais que:

$$\begin{cases} E^* + T^* = \Delta = \max_{i \leq j} [\sum_{k=i}^{j} p_k - (d_j - e_i)] \\ g(E^*) = h(T^*) \end{cases}$$

O algoritmo proposto nestes dois artigos é idêntico e composto

por duas fases. Numa primeira fase é determinada a permutação óptima das tarefas e na segunda fase são determinados os momentos óptimos de início das tarefas nessa permutação.

A partir da propriedade apresentada obtem-se a sequência óptima, atribuindo-se os momentos de início de cada tarefa (S_j) segundo a regra: $S_1 = e_1$ e $S_j = \max (e_j, C_{j-1})$ para $j = 2, \ldots, n$.

Dada uma sequência nestas condições, Lakshminarayan et al demonstram que Δ é igual ao atraso máximo. Desta forma, se Δ é não positivo os momentos correntes de início das tarefas são óptimos, já que o atraso e a antecipação máximos são nulos. Se Δ é estritamente positivo, então os momentos óptimos de início são dados pela diferença entre os momentos correntes de início e E^*.

Demonstrando-se que o custo da sequência assim obtida é igual ao minorante dado pelo lema, prova-se a sua optimalidade.

2.3.3.3 - Inserção de Tempo de Paragem na Máquina

Não constituindo por si só um problema de escalonamento de tarefas, o problema de inserção de tempos de paragem na máquina tem sido abordado em vários trabalhos.

O problema da inserção de tempo de paragem na máquina, que pode consistir na inserção de paragens entre o processamento de duas tarefas ou no começo do processamento da primeira tarefa depois do momento zero, é facilmente optimizável, dada uma permutação das tarefas, através de um problema de programação linear.

Este problema foi objecto de diferentes trabalhos de entre os quais se destacam os de Davis e Kanet (a) e b)) e de Fry, Leong e Rakes (1987).

Os trabalhos de Davis e Kanet têm uma relevância especial no contexto desta dissertação pois consideram uma configuração dos parâmetros idêntica à do *MCTDM*.

A optimização da inserção de tempos de paragem na máquina consiste na determinação das sequências de menor custo para cada permutação das tarefas. Uma sequência nessas condições diz-se sequência semi-activa.

Davis e Kanet apresentam um algoritmo que permite determinar as sequências semi-activas em tempo polinomial, dada uma permutação, S, das tarefas.

Este resultado permite reduzir a pesquisa da solução óptima ao conjunto π, das $n!$ permutações das tarefas.

Os resultados obtidos por Davis e Kanet permitem confirmar que a consideração de R2 torna o problema distinto do que é obtido sem a sua consideração. A versão restrita do *MCTDM* tem soluções óptimas distintas das da versão não restrita.

2.4 - Resolução por Enumeração Implícita

Na resolução optimal de problemas de optimização combinatória *NP-Difíceis* é usual a utilização de métodos de enumeração implícita, de que são exemplos algoritmos de tipo *branch-and-bound* e *programação dinâmica*.

Para tornar mais eficiente esta pesquisa é conveniente dispôr de regras de cancelamento da pesquisa ao longo de sub-sequências que se prove serem sub-optimais.

Estas regras baseiam-se, quer em propriedades das soluções óptimas que sejam formalizáveis como regras de dominância, quer no cálculo de minorantes e majorantes do valor da solução óptima do problema.

Nos trabalhos de investigação que têm recorrido à programação dinâmica foram usadas relaxações do espaço dos estados para a obtenção de minorantes para o valor da solução óptima do problema. São exemplos deste tipo de abordagens as realizadas em Abdul-Razaq e Potts (1988) e Miguel Constantino (1989).

Em algoritmos do tipo *branch-and-bound* o cálculo de minorantes tem sido realizado em diferentes trabalhos.

Fry, Leong e Rakes (1987), apresentam um teorema que permite calcular um minorante para o custo de sequenciar cada uma das tarefas ainda não sequenciadas, num problema em que as penalidades são idênticas para todas as tarefas e Ow e Morton (1989) utilizam uma relaxação do problema *MCTDM* num problema de afectação, como já havio sido proposto por Rinnooy Kan et al (1975) para outro problema de escalonamento de tarefas.

Uma outra forma de acelerar a pesquisa consiste na utilização de regras de dominância que permitam eliminar a pesquisa ao longo de algumas sub-sequências.

O estudo das características do *MCTDM* revela que dificilmente será possível determinar regras que permitam uma redução eficaz do espaço de pesquisa da solução óptima, em especial porque a função objectivo não tem um comportamento regular e porque as *due dates* são exógenas e diferentes para cada tarefa.

Os mais recentes trabalhos que propõem uma resolução optimal por enumeração implícita para os problemas *n/1/ /ET* com *due dates* distintas, utilizam regras de dominância específicas do problema que tratam, como em Fry et al (1987) em que as penalidades são idênticas para todas as tarefas. Não sendo estas regras generalizáveis ao *MCTDM*, tem sido mais vulgar a utilização de minorantes e majorantes na limitação do número de nodos a avaliar.

Das regras de dominância aplicáveis a este problema identificam-se duas que são independentes das características da função objectivo:

- **Teorema da Dominância da Programação Dinâmica** (*TDPD*): dadas duas sequências contendo os mesmos subconjuntos de tarefas é eliminada a pesquisa ao longo daquela que tenha o maior custo.

Apesar de potente, restrições de ordem computacional não permitem a utilização desta regra para análise de todas as sub-sequências de tarefas, e a sua utilização tem sido feita comparando o custo de sub-sequências de tarefas com o de majorantes do custo óptimo conhecidos.

Um caso paticular desta regra, e de mais fácil utilização, pode ser considerada:

- Regra de Intercâmbio de Tarefas Adjacentes (*AJI*): dadas duas sub-sequências de tarefas (i,j), dita corrente, e (j,i), se a penalidade associada à primeira for superior à da segunda então a pesquisa segundo a sub-sequência corrente pode ser cancelada.

Partindo da comparação do custo de sequenciação de tarefas adjacentes, tal como é realizado na regra *AJI*, obtem-se uma condição de adjacência de tarefas que, podendo ficar longe de garantir a optimalidade global, permite tornar mais eficiente a resolução do problema, quer em processos construtivos quer em processos de enumeração implícita.

Esta condição é apresentada por Ow e Morton (1989) para o problema *MCTDM* . Também Rachamadugu (1987) desenvolve regras do mesmo tipo para o problema com penalidades de antecipação nulas.

Através de argumentos de trocas de tarefas adjacentes, provam-se as condições de adjacência para todas as situações descritas na Tabela 1.

Condição de Adjacência:
Todos os pares de tarefas adjacentes numa sequência óptima devem satisfazer a seguinte condição:

$$b_i * p_j * \Theta_{ij} * (b_i + a_i) \leq b_j * p_i - \Theta_{ji} * (b_j + a_j)$$

em que a tarefa i precede imediatamente a tarefa j e Θ_{ij} e Θ_{ji} são definidas como:

$$\Theta_{lm} = \begin{cases} 0 & se\ f_l \leq 0 \\ f_l & se\ 0 < f_l < p_m \\ p_m & se\ f_l \geq p_m \end{cases}$$

em que $f_x = (T + p_x - d_x)$, é a folga da tarefa x se escalonada no período T.

A demonstração destas regras é bastante simples, através da utilização de argumentos de troca de tarefas, na medida em que a troca da ordem das tarefas *i* e *j* na sequência em nada altera o custo de sequenciação das restantes tarefas.

É importante notar que esta condição de optimalidade local se reduz às sub-sequências *Weighted Longest Processing Time (WLPT)* e *Weighted Shortest Processing Time (WSPT)* para as situações 1 e 9 da Tabela 1, que constituem o resultado já conhecido na Secção 2.3.3 na Propriedade III.

SITUAÇÃO	1	2	3	4	5	6	7	8	9
Antes da Troca (i,j)	(E,E)	(E,E)	(E,T)	(E,T)	(E,T)	(E,T)	(T,E)	(T,T)	(T,T)
Depois da Troca (i,j)	(E,E)	(T,E)	(E,E)	(E,T)	(T,E)	(T,T)	(T,E)	(T,E)	(T,T)

TABELA 1[2]- Situações de Adjacência relevantes para a condição apresentada

Refira-se para finalizar uma regra que é possível demonstrar e que permite reduzir a dimensão do problema fixando previamente algumas posições da sequência óptima: se num dado sistema de escalonamento há uma só tarefa *j* que tem *due date* superior à soma dos tempos de processamento de todas as tarefas do sistema, $d_j \geq \sum_{j \in N} p_j$, então essa tarefa é colocada na n-ésima posição da sequência.

[2][E] representa uma situação de escalonamento da tarefa antecipadamente em relação à sua *due date* e [T] uma situação de atraso.

3 - Métodos Heurísticos de Resolução do Problema *MCTDM*

Apresentam-se neste Capítulo os métodos heurísticos implementados para a resolução do *MCTDM*.

À semelhança do que é feito para a generalidade dos problemas de optimização combinatória que pertencem à classe *NP*, é apresentado um conjunto de métodos heurísticos para resolução do *MCTDM*, com os quais se pretendem obter bons resultados, em algoritmos que não necessitem da enumeração de um grande número de soluções.

Desta forma, põem-se em confronto as duas medidas do desempenho de um procedimento heurístico: a qualidade dos resultados e o esforço computacional (medido em tempo e em área de memória) necessário para os obter (Zanakis e Evans, 1981).

Os métodos estudados dividem-se em dois grupos, os de **tipo construtivo** e os de **tipo melhorativo**.

3.1 - Métodos Heurísticos Construtivos

Os **métodos heurísticos construtivos** partem de uma solução inicial vazia e vão passo a passo (em cada iteração dos algoritmos), construindo uma solução parcial. Desta forma, apenas no final do algoritmo é conhecida uma solução admissível para o problema.

Estes métodos baseiam-se em regras de escalonamento que, não garantindo a optimalidade, conduzem o processo de geração de soluções admissíveis para o problema de forma a que o valor da solução obtida esteja o mais próximo possível do da solução óptima.

A utilização de algoritmos construtivos, em que a sequenciação de uma tarefa numa dada posição tem um carácter irreversível, leva-nos a obter, frequentemente, soluções sub-optimais, provocadas por decisões de escalonamento incorretas e que não são corrigidas.

Para tentar melhorar a qualidade dos resultados obtidos com este tipo de regras é frequente adoptar algumas das seguintes medidas (Hart e Shogan (1987)): utilizar várias regras heurísticas para o mesmo problema; utilizar várias especificações matemáticas para as funções que conduzem a heurística; combinar várias heurísticas; perturbar os dados do problema e re-aplicar a heurística; introduzir aleatoriedade no processo heurístico.

Algumas destas medidas foram testadas relativamente às regras heurísticas utilizadas na resolução do problema *MCTDM*.

As regras consideradas neste trabalho, podem dividir-se em três grupos: **regras simples de escalonamento, regras de função prioridade** e **regras com antecipação do custo total da sequência.**

Os algoritmos construtivos implementados seguem o esquema apresentado na Figura 1.

```
ALGORITMO

Passo 1. Inicialização:
         1.1 Leitura da instância do problema
         1.2 Inicialização das variáveis e parâmetros

Passo 2. Repetir n vezes:
         2.1 Seleccionar tarefa a escalonar
         2.2 Actualizar variáveis do problema

Passo 3. Apresentar solução. FIM
```

FIGURA 1 - Esquema dos Algoritmos Construtivos

A Inicialização inclui as operações apresentadas na Figura 2.

```
1.1 Input: Número de Tarefas (n);
           Tempos de Processamento [p_i];
           Due dates [d_i];
           Penalidades de antecipação [a_i];
           Penalidades de atraso [b_i];
           Parâmetros do algoritmo;

1.2 Output: Posição [número de ordem na sequência];
            Término [momento de conclusão];
            FO (valor da sequência encontrada).

      T = 0 (momento corrente de escalonamento)
      S = (1,...,n) (conjunto de tarefas por escalonar)
      S' = () (conjunto de tarefas já escalonadas)
      FO = 0 (valor corrente da função objectivo)
      IT = 1 (iteração corrente)
```

FIGURA 2 - Passo 1 dos Algoritmos Construtivos

Os algoritmos implementados distinguem-se no critério de selecção da tarefa a escalonar, pelo que o Passo 2.1 será apresentado separadamente para cada regra.

No Passo 2.2 a actualização da Função Objectivo (FO) é feita adicionando-se à FO corrente a penalização associada ao escalonamento da tarefa seleccionada e esta penalização é calculada através da seguinte função:

$$\Delta FO(l,T) = \begin{cases} a_l * (d_l - T - p_l) & se\ d_l > T \\ b_l * (T + p_l - d_l) & se\ d_l \leq T \end{cases}$$

A actualização das variáveis consiste na realização do conjunto de operações apresentadas na Figura 3.

```
     Seja l a tarefa a escalonar na
     iteração corrente:
2.2  S = S - { l }
     S' = S' + { l }
     POSIÇÃO[l] = IT

     FO = FO +  Δ FO(l,T)

     T = T + p_l
     TERMINO[l] = T
     IT = IT + 1
```

FIGURA 3 - Passo 2.2 dos Algoritmos Construtivos

3.1.1 - Algoritmos Baseados em Regras Simples de Escalonamento

Com os algoritmos baseados em regras simples de escalonamento pretende-se tirar partido de algumas regras de escalonamento que resolvem optimalmente casos particulares do *MCTDM*, apresentadas no Capítulo 2.

Uma das características destes algoritmos é a de permitirem a obtenção de soluções com uma grande rapidez de cálculo, pois as regras de escalonamento utilizadas baseiam-se, para cálculo do indicador que define a tarefa a escalonar, apenas na informação específica relativa a cada tarefa, tal como: *due dates*, tempos de processamento, penalidades de antecipação, etc.

Existe um grande número de regras heurísticas simples, cada uma com especificidades próprias na aplicação a diferentes problemas. Panwalkar e Iskander (1977) referenciam mais de 100 regras deste tipo.

As regras simples de escalonamento utilizadas podem classificar-se do seguinte modo (Panwalkar e Iskander, (1977)):

a) Regras Relacionadas com as *due dates*;
b) Regras Relacionadas com as penalidades;

c) Regras Relacionadas com as penalizações (conjugação de informação sobre as *due dates* e sobre as penalidades);

3.1.1.1 - Regras Relacionadas com as *Due Dates*

Desta classe foi implementada uma heurística baseada na regra que consiste em seleccionar a tarefa com menor *due date* (*EDD*) de entre as ainda não escalonadas.

São de esperar resultados razoáveis com esta regra em problemas em que as *due dates* estejam bastante dispersas, situação em que os desvios entre os momentos de conclusão e as *due dates* tenderão a ser menores, pelo que a não inclusão das penalidades no critério de escalonamento terá um menor impacto ao nível dos resultados. Um algoritmo baseado nesta regra foi utilizado como termo de comparação com outras heurísticas, para a resolução do *MCTDM*, por Ow e Morton (1989) e por Davis e Kanet (a) e (b).

É de especial importância a sua utilização no problema de minimização da penalidade máxima, em que a permutação resultante de uma ordenação das tarefas por ordem *EDD* constitui a permutação óptima (Sidney (1977) e Lakshminarayan et al (1977)).

- Ordem Crescente de *Due Date* (EDD)

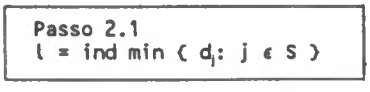


FIGURA 4 - Passo 2.1 do Algoritmo EDD

3.1.1.2 - Regras Relacionadas com as Penalidades

Foi desenvolvido um algoritmo baseado numa regra de sequenciação que na decisão de qual a tarefa a escalonar apenas toma em consideração as penalidades por unidade de tempo de processamento

(a_j/p_j e b_j/p_j).

Utilizam-se neste algoritmo as regras de escalonamento conhecidas como Ordem decrescente de penalidade de atraso por unidade de tempo de processamento (*WSPT*) e Ordem crescente de penalidade de antecipação por unidade de tempo de processamento (*WLPT*).

Estas regras de sequenciação constituem os dois ramos do V-invertido, que resultam da Propriedade III apresentada no Capítulo 2. Dado um conjunto de tarefas adiantadas, estas deverão ser sequenciadas por ordem *WLPT*, situação 1 da condição de adjacência (Tabela 1) e dado um conjunto de tarefas atrasadas estas deverão ser sequenciadas por ordem *WSPT*, situação 9 da regra de adjacência (Tabela 1).

Intuitivamente será de esperar que os resultados produzidos com esta regra sejam de melhor qualidade em situações de fraca dispersão das *due dates*, já que esta situação corresponde a ter grande parte das tarefas, simultaneamente, atrasada ou antecipada em relação às respectivas *due dates*.

Assim esta regra tenderá a produzir melhores resultados em instâncias com características opostas àquelas em que se espera que a regra *EDD* venha a ter um melhor comportamento.

No algoritmo implementado, num primeiro momento é avaliado o número de tarefas por escalonar que, se escalonadas na posição corrente, terminariam adiantadas em relação à respectiva *due date* e o número das que terminariam atrasadas. Em seguida selecciona-se a tarefa a escalonar pela regra *WLPT*, se o primeiro for superior ao segundo, e pela regra *WSPT* no caso contrário.

■ Ordem crescente (ou decrescente) de penalidade de antecipação (ou atraso) por unidade de tempo de processamento (*WSWL*)

```
C1 = 0
C2 = 0
∀ j ∈ S,
Se dj > (T + pj) então
  C1 = (C1 + 1)
Caso contrário
  C2 = (C2 + 1)
Se C1 > C2 então
  l = ind min ( aj/pj: j ∈ S )
Caso contrário
  l = ind max ( bj/pj: j ∈ S )
```

FIGURA 5 - Passo 2.1 do Algoritmo WSWL

3.1.1.3 - Regras Relacionadas com as Penalizações

As regras *EDD* e *WSWL* não consideram, cada uma delas, um aspecto importante dos dados do problema. A primeira não tem em conta as penalidades em que se incorre e na segunda apenas indirectamente se consideram as *due dates*.

Com esta terceira regra pertende-se combinar a informação relativa às penalidades e às *due dates*, sequenciando-se em cada iteração do algoritmo a tarefa que menos contribui para o aumento do valor da sequência.

O carácter *greedy* desta regra deve-se a que em cada iteração o escalonamento proposto é o da tarefa com a qual se verifica o menor incremento no valor da função objectivo.

■ Mínimo da Função Custo (FC)

```
∀ j ∈ S,
Calcular  Δ FO(j,T)

l = arg min ( Δ FO(j,T): j ∈ S )
```

FIGURA 6 - Passo 2.1 do Algoritmo FC

3.1.2 Algoritmos Baseados em Funções de Prioridade

A ideia base do desenvolvimento deste tipo de algoritmos é a de que, não sendo conhecidas regras simples que garantam a optimalidade global, deveremos aplicar regras de escalonamento baseadas no cálculo de funções prioridade que, procurando ultrapassar as características mais difíceis do problema, conjugam alguns resultados conhecidos localmente, com medidas da prioridade de escalonamento de cada tarefa que permitam antever a aproximação de situações de congestão de *due dates* de diferentes tarefas.

Esta última característica parece ser a mais difícil de enfrentar em processos construtivos, mesmo que sejam garantidas algumas condições de adjacência de tarefas numa sequência. Numa situação de congestão de *due dates*, um dado conjunto de tarefas poderá passar de uma situação de adiantamento em relação à respectiva *due date*, para uma situação de atraso.

Perante uma situação deste tipo, o recurso a condições de adjacência, baseadas na troca de duas tarefas adjacentes, num problema em que estas condições dependem da subsequência que antecede essas tarefas, pode levar-nos a soluções globalmente de má qualidade ainda que possam verificar, duas a duas, algum critério de optimalidade local (definido com base em critérios de adjacência).

É da capacidade que uma determinada função de prioridade tem de antever as situações de congestão de *due dates*, que depende, em boa medida, a qualidade dos procedimentos heurísticos construtivos que a ela recorram.

Na definição da prioridade, as funções implementadas utilizam os critérios de adjacência apresentados na secção 2.4, para algumas situações particulares de antecipação e atraso das tarefas.

Pretende-se, no entanto, que a pesquisa do espaço das soluções tenha em vista algo mais do que condições de adjacência, introduzindo-se alguma capacidade de antecipação das situações de aglomeração de *due dates*.

Foram implementadas três alternativas para a função de prioridade que conduz a heurística, uma especificação linear e uma exponencial, baseadas nas conclusões dos trabalhos de Ow e Morton (1989), que fizeram uma implementação similar, e é proposta uma terceira formulação matemática da função de prioridade que se baseia num cálculo do custo local da sequência.

As funções propostas por Ow Morton têm as seguintes expressões:

$$PR_j^L = \begin{cases} B_j & se\ F_j \le 0 \\ B_j - F_j * \dfrac{(B_j + A_j)}{k * P} & se\ 0 < F_j \le k * P \\ -A_j & se\ F_j > k * P \end{cases}$$

FIGURA 7 - Função Prioridade Linear

$$PR_j^E = \begin{cases} B_j & se\ F_j < 0 \\ B_j * e^{\left(\frac{-(A_j + B_j)}{A_j} * \frac{F_j}{P}\right)} & se\ 0 \le F_j \le \dfrac{B_j}{(A_j + B_j)} * k * P \\ a_j^{-2} * \left(B_j - \dfrac{(A_j + B_j) * F_j}{k * P}\right)^3 & se\ \dfrac{B_j}{(A_j + b_j)} * k * P < F_j < k * P \\ -A_j & se\ F_j > k * P \end{cases}$$

FIGURA 8 - Função de Prioridade Exponencial

Em que $\bar{P} = \frac{1}{n} \sum_{j \in N} p_j$, $F_j = (d_j - T - p_j)$ (folga da tarefa j em relação à sua *due date*), $B_j = b_j/p_j$, $A_j = a_j/p_j$, T é o momento corrente de escalonamento e k é um parâmetro que permite reduzir a "miopia" da heurística em relação a situações de pouca dispersão das *due dates*.

A terceira implementação da função de prioridade baseia-se na ideia de inserir na sequência a tarefa que, não sendo inserida no período corrente, apresente uma maior diferença entre o custo da sua sequenciação imediata e o custo da melhor sequenciação na iteração seguinte (uma visão "optimista" da sua melhor sequenciação na iteração seguinte, ou seja, do seu custo localmente definido).

O valor da "prioridade" seria assim medido através da diferença entre o custo Δ FO(j,T) do escalonamento no período corrente e o escalonamento de menor custo possível para a tarefa j na posição/iteração seguinte, dado o conjunto das tarefas que estão por escalonar.

A expressão matemática da função prioridade assim definida foi alterada no sentido de incorporar o conhecimento adquirido sobre a optimalidade de algumas sequências parciais em que todas as tarefas estão atrasadas ou antecipadas em relação à sua *due date*.

A função prioridade que se propõe é a que se segue:

$$PR_j^A = \begin{cases} -A_j * p_{max} * \theta & se \;\; O_j > T + \max\limits_{k \in S \setminus j} \{p_k\} \\ H_j - A_j * (d_j - T - p_j) & se \;\; T \leq O_j \leq T + \max\limits_{k \in S \setminus j} \{p_k\} \\ B_j * p_{min} & se \;\; O_j < T \end{cases}$$

FIGURA 9 - Função de Prioridade com Cálculo do Custo Local (ACL)

Em que se define $O_j = d_j - p_j$ como sendo o momento óptimo de início da tarefa j. Os valores de p_{max} e p_{min} são, respectivamente, o tempo de processamento da tarefa de maior e menor duração de entre as que ainda não foram escalonadas, excepto a tarefa j, e θ é um parâmetro que permite distinguir os níveis de prioridade nos primeiro e segundo ramos da função.

Finalmente,

$$H_j = \min \{A_j * (d_j - T - p_j - p_k),\ B_j * (T + p_j + p_l - d_j)\}$$

em que,

$$k = \arg\min_{x \in M} \{ \Phi(x) \}, \quad M = \{h \in S: \ \Phi(h) \geq 0 \}$$

$$l = \arg\min_{x \in m} \{ \Phi(x) \}, \quad m = \{h \in S: \ \Phi(h) < 0 \}$$

$$\Phi(x) = T + p_j + p_x - d_j \qquad x \in N$$

ou seja, k é a tarefa de S que, caso exista, permite a j ficar o menos adiantada possível se for escalonada imediatamente a seguir a k; l é a tarefa de S que, caso exista, permite a j ficar o menos atrasada possível se for escalonada imediatamente a seguir a l[3]

Os algoritmos implementados seguem o seguinte esquema:

[3]Na implementação computacional do algoritmo, quando para uma dada tarefa j não existem tarefas k ou l nas condições apresentadas, foram considerados, por defeito, tempos de processamento suficientemente grandes para duas tarefas artificiais de forma a que a situação de melhor antecipação ou de melhor atraso não se tornasse elegível pelo algoritmo.

■ Máximo da Função Prioridade (*LIN-ET*, *EXP-ET*, *ACL*)

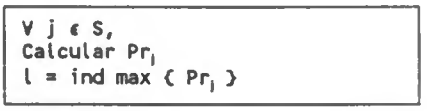


FIGURA 10 - Passo 2.1 dos Algoritmos LIN-ET, EXP-ET e ACL

3.1.3 Algoritmos Baseados em Regras com Antecipação do Custo Total da Sequência

Os algoritmos com avaliação do custo total da sequência caracterizam-se por basearem a decisão sobre a tarefa a escalonar, numa prospecção do custo final das soluções obtidas com o seu escalonamento na posição corrente.

Estes algoritmos são compostos por duas fases, numa primeira é avaliada a prioridade do escalonamento imediato de cada uma das tarefas por escalonar e na segunda é calculado um majorante para o valor da sequência que resulta de escalonar cada uma das tarefas em análise na posição/iteração corrente.

Na primeira fase do algoritmo o indicador da prioridade de escalonamento utilizado é o que resulta do cálculo de uma das funções apresentadas na secção 3.1.2.

Para cálculo dos majorantes, na segunda fase, consideram-se as tarefas cujo valor da função prioridade seja inferior em menos de *p*% ao melhor, limitando-se assim esse cálculo ao conjunto de tarefas cujo escalonamento imediato pareça mais promissor, reduzindo-se o esforço computacional efectuado, que se pode tornar excessivo para problemas de dimensão muito grande.

Em cada iteração dos algoritmos é sequenciada a tarefa para a qual for calculado um menor majorante do custo total da

sequência.

O número de subsequências que é necessário avaliar não é constante, crescendo com a proximidade dos valores que se obtiverem para a função prioridade.

Este procedimento é uma simplificação do realizado por Ow e Morton (198), em que é calculado o majorante para um número fixo de tarefas em cada iteração e é construído, em paralelo, um número de sequências completas não necessariamente igual a um. Desta forma o procedimento de Ow e Morton assemelha-se a uma árvore truncada e sem retrocesso na pesquisa, enquanto que o procedimento aqui descrito consiste num aprofundamento da pesquisa em algoritmos construtivos.

A eficácia deste tipo de métodos depende da qualidade dos dois procedimentos nele envolvidos. A função prioridade determina as subsequências que são sujeitas a avaliação, excluindo dessa avaliação as de baixa prioridade. A tarefa a escalonar de imediato é determinada em função do valor do majorante usado como instrumento de avaliação. A obtenção de uma solução de boa qualidade com pouco esforço computacional depende da capacidade da função prioridade para filtrar as melhores subsequências e da capacidade do procedimento usado na determinação do majorante de identificar a mais favorável.

Foram implementadas nove variantes para esta heurística, que resultam da conjugação das três funções de prioridade com as três regras simples de escalonamento apresentadas.

O algoritmo considerado segue o seguinte esquema:

■ Antecipação do Custo Total da Sequência (ACTS)

$\forall j \in S$,
Calcular Pr_j
$i = \text{ind max} \{ Pr_j \}$
$Pr^M = Pr_i$
Seja, $P_s = \{j: j \in S \text{ e } Pr_j \geq (1-p) * Pr^M\}$
$\forall j \in P_s$
Calcular Ub_j,
$l = \text{ind min} \{ Ub_j \}$
$Ub_m = Ub_l$

FIGURA 11 - Passo 2.1 dos Algoritmos ACTS

3.2 - Métodos Heurísticos Melhorativos Determinísticos

Os **métodos heurísticos melhorativos** são muitas vezes conhecidos como técnicas de pesquisa local ou de melhoramentos iterativos (Johnson et al, 1989).

Neste tipo de procedimentos procura-se melhorar uma solução inicial, através da realização de pequenas alterações, definidas localmente. Este processo decorre até que nenhuma dessas alterações permita obter uma solução de valor inferior ao da solução corrente, solução essa que constitui um óptimo local.

Os algoritmos melhorativos surgem como uma tentativa de corrigir as decisões erradas não corrigidas nos algoritmos construtivos.

Enquanto que a pesquisa do espaço das soluções feita em métodos construtivos, tem por base sequências parciais, a pesquisa em métodos melhorativos tem por base sequências completas.

3.2.1 - Pesquisa Local

Genericamente um algoritmo de pesquisa local pode ser descrito pelo seguinte conjunto de passos:

```
Passo 1. Encontrar uma solução inicial S
Passo 2. Enquanto houver uma solução vizinha de S, S':
         Custo(S') < Custo(S)  fazer S=S'
Passo 3. Apresentar solução final S.
```

FIGURA 12 - Algoritmo Genérico de Pesquisa Local

Na aplicação de um algoritmo de pesquisa local a um determinado problema de optimização combinatória, torna-se necessário

responder a um conjunto de questões específicas do problema. O sucesso da aplicação depende, em parte, das respostas que forem dadas a essas questões.

Um bom trabalho de síntese sobre os algoritmos de pesquisa local pode ser encontrado, por exemplo, em Papadimitriou e Steiglitz (1982).

Uma questão central na definição de uma aplicação do algoritmo genéricamente apresentado é a escolha da vizinhança em que se definem os óptimos locais.

A vizinhança de uma solução constitui o sub-espaço das soluções em que decorre o Passo 2 do algoritmo de pesquisa local.

A qualidade do óptimo local encontrado está, muitas vezes, dependente da dimensão desse sub-espaço. É de esperar que a escolha de vizinhanças de maior dimensão proporcione a obtenção de melhores óptimos locais embora à custa de um maior esforço computacional.

No *MCTDM* a dimensão da vizinhança de uma solução admissível está dependente de dois factores: do número de tarefas cujas posições se admite alterar e da forma como se definem as alterações a considerar.

Na implementação feita, apenas são consideradas trocas de posições entre duas tarefas adjacentes na solução corrente.

O conjunto V_S das sequências vizinhas de uma sequência S é:

V_S = {S': S' é obtido de S por troca das posições de duas quaisquer tarefas adjacentes}

Esta definição de vizinhança coincide com a noção adiantada na Secção 2.4 e que permitiu definir o critério de adjacência aí apresentado.

A utilização deste tipo de vizinhanças permitirá analisar a importância que a estrutura de uma dada solução tem na sua qualidade, localmente definida, e a capacidade que as vizinhanças utilizadas têm em melhorar soluções iniciais construídas com estruturas distintas.

Um outro procedimento a definir consiste na forma como se realiza a pesquisa na vizinhança de uma dada solução.

A ordem de pesquisa das soluções pertencentes à vizinhança é uma das questões a definir neste procedimento.

Nesta implementação as tarefas são ordenadas segundo as posições que ocupam na sequência corrente e podem originar trocas com a tarefa adjacente que está na posição imediatamente a seguir.[4]

As sucessivas trocas testadas são conduzidas de forma circular, em oposição a testar trocas em pontos aleatórios da sequência.

Da mesma forma, quando há mudança de solução, a pesquisa na vizinhança da nova solução é iniciada no ponto em que a anterior tinha terminado, mantendo-se, também aqui o carácter circular da pesquisa.

Ao longo de uma mesma solução a pesquisa continua até que é encontrada uma primeira troca melhorativa, situação em que essa troca é realizada, mudando-se imediatamente de solução.

Para completar a descrição da forma como a pesquisa nas vizinhanças é feita, falta apenas referir a forma como se inicia a pesquisa na vizinhança de uma dada solução inicial.

Consideraram-se como primeiras soluções a pesquisar na vizinhança da solução inicial, as que correspondem à troca da tarefa com maior contribuição para a função objectivo com as sequenciadas

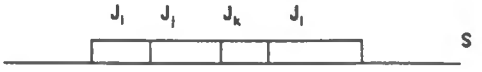
---

[4]Considerou-se que a posição N não é adjacente à posição 1, pelo que as tarefas sequenciadas nessas posições apenas têm uma tarefa adjacente e a tarefa N não dá origem a qualquer troca na vizinhança.

em posições adjacentes (quando estas existam).

A troca inicialmente realizada é a melhor das que correspondam a um decréscimo da função objectivo. Caso não se verifique a existência de trocas melhorativas a pesquisa prossegue na mesma solução.

O seguinte exemplo pretende descrever a forma como a pesquisa é realizada. Seja a seguinte sub-sequência de tarefas de uma dada sequência S:



e a solução na sua vizinhança obtida por troca das tarefas J_j e J_k.

Suponha-se que a troca é realizada, isto é, Custo(S') < Custo (S), então a próxima troca a ser testada, na vizinhança de S' é a da tarefa J_i com a tarefa J_k. Caso a troca não tivesse sido realizada a solução corrente, S, manter-se-ía e a troca seguinte a testar seria a da tarefa J_k com J_i.

Este tipo de pesquisa parece ser mais eficiente do que entrar na vizinhança de cada nova solução pelo método proposto para a solução inicial, já que a troca de duas quaisquer tarefas J_j e J_k não altera o custo de sequenciação das tarefas colocadas em posições anteriores a J_j e posteriores a J_k, eliminando-se assim, um considerável número de análises repetitivas de trocas não vantajosas.

Para completar a descrição da aplicação feita do algoritmo de pesquisa local é necessário definir uma forma de obtenção da solução inicial (Passo 1).

No geral, nas aplicações de pesquisa local, consideram-se várias soluções de partida. Este procedimento justifica-se pelo facto de, nos problemas de optimização combinatória, se poder verificar a existência de vários óptimos locais, pelo que existe o risco de se ficar preso a óptimos locais de má qualidade.

Neste sentido as soluções iniciais foram obtidas através de quatro regras de escalonamento distintas: uma solução obtida através da regra com função de prioridade linear (*LIN-ET*), uma outra utilizando a função *ACL*, uma solução dada pela regra simples de escalonamento *WSWL* e uma quarta pela regra *EDD*.

O algoritmo implementado é o seguinte:

- Pesquisa Local

```
1. INICIALIZAÇÃO
   1.1 Ler a instância do problema
   1.2 Determinar uma Solução Inicial, S'
   1.3 Solução Corrente = S = S'
   1.4 CONT = 0 (Contador do número de trocas consecutivas testadas e
                 não realizadas)

2. TROCA INICIAL

   2.1 Determinar l = arg max ( Δ FO(j): j ∈ N )

   2.2 Determinar melhor solução na vizinhança de l, S' e ir para 3.1.2


3. OBTENÇÃO DE UM ÓPTIMO LOCAL
   Repetir enquanto CONT < n
   3.1 Seja S' a solução na vizinhança de S a considerar
       3.1.1 Calcular Custo(S')
       3.1.2 Calcular DELTA = Custo(S') - Custo(S)
       3.1.3 Se DELTA < 0 ir para 3.1.5
       3.1.4 CONT = (CONT + 1) e ir para 3
       3.1.5 [Realizar a troca:]
             CONT = 0
             Custo(S) = Custo(S')
             S = S'
4. APRESENTAR SOLUÇÃO FINAL.
```

FIGURA 13 - Algoritmo de Pesquisa Local

3.3 - Métodos Heurísticos Melhorativos Aleatorizados

Nos algoritmos de pesquisa local apresentados na Secção 3.2 todas as decisões são tomadas de forma determinística, ou seja, aceitam-se com **probabilidade um** as trocas melhorativas e com **probabilidade zero** todas as trocas não melhorativas.

Não se conhecendo propriedades das soluções óptimas de que se possa tirar partido para a pesquisa da solução óptima, esta rigidez pode levar a que, ainda que garantindo a obtenção de um óptimo local, o processo de pesquisa não detecte óptimos locais de qualidade superior só porque para ter acesso a essas soluções partindo da solução inicial adoptada é necessária a aceitação de transições não melhorativas.

Uma forma de tentar reduzir o risco de se ficar preso a óptimos locais deste tipo consiste na aplicação de métodos em que, de forma controlada, se aceitam trocas não melhorativas, como são os do tipo *Simulated Annealing (S.A.)* , desenvolvidos a partir dos trabalhos de Kirkpatrick et al (1983) e Cerny (1985).

Uma outra classe de algoritmos com os mesmos principios são sintetisados no trabalho de Glover (1989) e são denominados algoritmos *Tabu Search (T.S.)*.

Com o auxílio da aleatorização controlada da pesquisa e a quebra de alguns dos princípios da pesquisa local, como o de apenas se aceitarem trocas melhorativas, tem-se tentado, com sucesso nalgumas aplicações, ultrapassar as principais limitações da pesquisa local tradicional (Johnson et al, 1989 e 1991).

A aplicação de algoritmos *S.A.* e *T.S.* a problemas de escalonamento de tarefas é referenciada em Storer, Wu e Vaccari

(1992) e em Van Laarhoven, Aarts e Lenstra (1992), que fazem uma aplicação de um algoritmo *S.A.* a um problema de *Job Shop*.

3.3.1 - Algoritmo Simulated Annealing

O Algoritmo Simulated Annealing que a seguir se descreve é baseado nas principais conclusões das aplicações feitas por Johnson et al (1989) a vários problemas de optimização combinatória.

O *S.A.* utiliza uma analogia com o processo físico do "annealing" para permitir de forma controlada alterações da solução corrente que aumentem o valor da função objectivo.

Uma apresentação simplificada do *S.A.* como consistindo num conjunto de dois ciclos é apresentada em Malek et al (1990):

```
Passo 1. Obter Solução Inicial
Passo 2. [Ciclo 1] Enquanto não se atingir o critério de paragem fazer
         2.1 [Ciclo 2] Enquanto não se atingir o equilíbrio
             2.1.1 Propor nova troca na solução corrente
             2.1.2 Se a troca é aceite alterar solução corrente
         2.2 Actualizar temperatura
Passo 3. Apresentar melhor solução obtida.
```

FIGURA 14 - Esquema Simplificado de Funcionamento do Simulated Annealing

O ciclo interno (Ciclo 2) decorre até que se atinge um ponto de equilíbrio, i.e., para cada temperatura é proposto um conjunto de trocas cuja aceitação depende de uma função de aceitação (ver Figura nº 15), considerando-se o ponto de equilíbrio atingido quando para essa temperatura é proposto um número de transições limite ou as transições aceites atingem um dado número, números esses que são previamente fixados (Passo 2.1 no algoritmo descrito detalhadamente na Figura 16).

Se a troca é aceite então ela é aplicada na solução corrente, gerando-se uma nova solução.

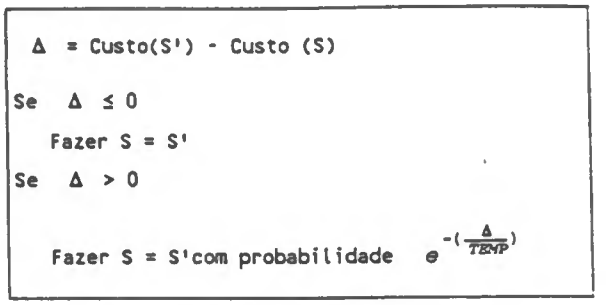


FIGURA 15 - Função de Aceitação das Trocas Propostas

O ciclo exterior (Ciclo 1) é repetido até que se atinge o critério de paragem, que corresponde ao critério de congelação na analogia com o processo físico de "annealing".

De cada vez que o ciclo interno é completado há uma redução da temperatura, avançando-se no sentido da congelação se a percentagem de trocas aceites for menor que um parâmetro previamente fixado.

A aplicação de um algoritmo *S.A.* a um determinado problema de optimização combinatória requer a resposta a um conjunto de questões, que se podem dividir em dois grupos, as que são específicas do problema e as que são genéricas em termos do processo de *annealing*.

Questões específicas do problema:

1. Definição de solução.
2. Definição da vizinhança de uma solução.
3. Fórmula de cálculo do custo de uma solução.
4. Processo de determinação da solução inicial.
5. Processo de determinação de uma solução na vizinhança e sua avaliação.

Questões genéricas do algoritmo:

1. Fixação da temperatura inicial ($TEMP_0$).
2. Fixação do coeficiente de redução da temperatura (*TEMPFACTOR*).

3. Fixação do número máximo de transições a gerar a uma mesma temperatura.
4. Definição do estado de congelação (*FREEZELIM*).

A descrição da aplicação do *S.A.* ao *MCTDM* vai ser feita respondendo às questões colocadas.

Foram mantidas as definições de solução, vizinhança e custo de uma solução, pelo que neste ponto são analisados os processos de determinação da solução inicial e de uma solução na vizinhança.

Para a obtenção de uma solução inicial foram testados dois procedimentos, através de uma heurística construtiva e através da geração de uma sequência por um método pseudo-aleatório. Os maus resultados obtidos com a utilização das sequências pseudo-aleatórias levaram ao seu afastamento como solução inicial.

A transição numa vizinhança é feita através da escolha, de forma pseudo-aleatória, de uma tarefa a partir da qual irá ser gerada a nova solução (S'), na vizinhança da anterior. Definida essa tarefa, podem ocorrer três situações: a tarefa está na posição *1*, podendo apenas trocar com a que está imediatamente a seguir (posição *2*); a tarefa está na posição *n* pelo que só pode trocar com a da posição (*n-1*); a tarefa está numa posição *m*, diferente de *1* e *n*, caso em que a troca proposta é com a tarefa adjacente com que se obtem uma solução de menor custo.

O algoritmo implementado segue o esquema representado na Figura 16 (é utilizada, sempre que possível, a denominação das variáveis e dos parâmetros do algoritmo apresentado em Johnson et al (1989)).

As questões genéricas ligadas ao processo de *annealing* dada a sua natureza são apresentadas no ponto dedicado à experiência computacional.

■ Simulated Annealing

```
1. INICIALIZAÇÃO
   1.1 Ler a instância do problema
   1.2 Fixar um valor inicial para a temperatura, TEMP0 > 0
   1.3 Fixar um valor para o coeficiente de redução da temperatura, 0 < TEMPFACTOR < 1
   1.4 Determinar uma Solução Inicial,  S
   1.5 Melhor Solução = S* = S

2. OBTENÇÃO DE UMA SOLUÇÃO FINAL
   Repetir enquanto não é atingida a temperatura de congelação
   (Freezecount < Freezelim)
   2.1 Repetir enquanto nao se atinge número máximo de transições
       (Trials < Sizefactor * Viz)
        e enquanto número de transições aceites for admissível
       (Changes < Cutoff * Viz)
       2.1.0 Trials = (Trials + 1)
       2.1.1 Determinar aleatoriamente solução admissível S' na vizinhança de S
       2.1.2 Calcular Custo(S')
       2.1.3 Calcular DELTA = Custo(S') - Custo(S)
       2.1.4 Se DELTA < 0 ir para 2.1.7
       2.1.5 Calcular P = exp (- DELTA * TEMP)
             Gerar um número pseudo-aleatório 0 ≤ x ≤ 1
       2.1.6 Se P < x rejeitar a transição ir para 2.1.0
       2.1.7 [Aceitar a transição:]
             Changes = Changes + 1
             Se Custo(S') > Custo(S*) ir para 2.1.8
                [Actualizar melhor solução:]
                Custo(S*) = Custo(S'), S* = S'
                Freezecount = 0
       2.1.8 [Actualizar solução corrente:]
             Custo(S) = Custo(S')
             S = S'
             Ir para 2.1.0
   2.2 Reduzir a temperatura TEMP = ( TEMPFACTOR * TEMP)
       Se Changes/Trials < MINPERCENT fazer
          Freezecount = Frezecount + 1
   2.3 Se Freezecount ≥ Freezelim ir para 3 caso contrário ir para 2.1

3. APRESENTAR SOLUÇÃO FINAL.
```

FIGURA 16 - Algoritmo Simulated Annealing

3.4 Aleatorização do Processo Construtivo

Incluem-se nesta secção os algoritmos em que foi aleatorizado o processo de construção da sequência de tarefas a partir dos algoritmos apresentados nas secções 3.1.2 e 3.1.3.

Os métodos construtivos apresentados procedem de forma iterativa à sequenciação das tarefas, seleccionando em cada iteração, no conjunto das tarefas ainda não escalonadas, uma tarefa para ser escalonada no momento corrente.

Estes algoritmos consideram, em cada momento, apenas a melhor opção avaliada, eliminando todas as outras, pelo que eventuais decisões tomadas erradamente são irreversíveis, afastando a solução em construção do conjunto das melhores soluções.

Torna-se intuitivo que uma das formas de contornar estas situações é a de seleccionar, não a tarefa com melhor indicador mas uma das que tenha melhor valor.

Com base neste raciocínio foram desenvolvidos vários algoritmos construtivos aleatorizados para o *MCTDM*.

Se por um lado se pretende testar a capacidade da aleatorização controlada das decisões de escalonamento, tomadas em cada iteração do método, ultrapassar os inconvenientes decorrentes de decisões locais globalmente incorrectas, por outro lado este tipo de procedimento aleatorizado constitui uma forma de pesquisar o espaço das soluções, em processos construtivos, não dependente da definição e características de espaços de vizinhanças.

São conhecidas aplicações de algoritmos deste tipo para problemas de optimização de rotas (Hart e Shogan, 1987), mas não para

problemas de escalonamento de tarefas.

A adaptação destes procedimentos ao problema de escalonamento de tarefas leva-nos a escolher para tarefa a sequenciar na iteração corrente, não a tarefa com melhor valor da função que comanda a heurística, mas uma escolhida aleatoriamente de entre as c melhores ou de entre as que não tenham um valor mais que $p\%$ diferente do da melhor (conjunto P_s).

Atendendo às características do problema em presença, e a que se espera que o valor da função prioridade e/ou dos majorantes calculados venha a assumir, em determinadas instâncias e num determinado momento do processo de escalonamento, valores bastante próximos, adoptou-se a alternativa que inclui no conjunto P_s as tarefas que não tenham valor da função de avaliação mais que $p\%$ distante do melhor valor.

Foram aplicados procedimentos deste tipo aos algoritmos baseados em funções prioridade e em regras com antecipação do custo total da sequência.

3.4.1 - Aleatorização dos Algoritmos Baseados em Funções Prioridade

Nos algoritmos baseados em funções prioridade foi aleatorizada a escolha da tarefa a sequenciar entre o conjunto das tarefas cuja prioridade não seja mais que $p\%$ inferior à prioridade máxima. Deste modo e atendendo a que se espera que a existência de valores muito próximos da função prioridade para diferentes tarefas seja um indicador de eventuais decisões alternativas que a função não consegue distinguir parece ser intuitivamente mais eficaz fazer depender o número de tarefas que se incluem em P_s dessas situações de indecisão.

Foram assumidas probabilidades idênticas para o escalonamento de todas as tarefas pertencentes a P_s e foram realizadas diversas corridas dos algoritmos.

O esquema seguido pelos algoritmos é o seguinte:

- Algoritmo Aleatorizado Baseado em Funções Prioridade (*PRIALEAT*)

```
P_e = ( ) (conjunto de tarefas seleccionadas na
          iteração corrente)
∀ j ∈ S,
Calcular Pr_j
i = ind max ( Pr_j )
Pr^M = Pr_i
Seja P_e = (j: j ∈ S e Pr_j ≥ (1-p) * Pr^M)
Escolher aleatoriamente l ∈ P_e
```

FIGURA 17 - Passo 2.1 dos Algoritmos LINET_A, EXPET_A e ACL_A

3.4.2 - Aleatorização dos Algoritmos Baseados em Regras com Antecipação do Custo Total da Sequência

Os algoritmos com antecipação do custo total da sequência são compostos por duas fases, uma primeira em que é calculada a prioridade de escalonamento para cada tarefa ainda não escalonada e a segunda em que são calculados majorantes para o custo da sequência que resulta de sequenciar cada uma das tarefas no momento corrente.

Nestes algoritmos foi aleatorizado apenas o processo de selecção da tarefa a escalonar, após o cálculo dos majorantes do custo da sequência, mantendo-se idêntico o processo de escolha das tarefas para os quais é calculado o majorante.

A tarefa escalonada é escolhida de entre aquelas para as quais foi calculado um majorante não superior em $s\%$ ao menor (que constituem o conjunto P_b). Também aqui se optou pela regra que faz variar o número de tarefas envolvidas no processo de escolha aleatorizada.

A probabilidade de selecção é idêntica para cada uma das tarefas pertencentes a P_b e foram realizadas várias corridas de cada algoritmo.

O algoritmo segue o seguinte esquema:

- Algoritmo Aleatorizado Baseado em Regras com Avaliação Antecipada do Custo da Sequência (*ACTSALEAT*):

P_a = ( ) (conjunto de tarefas seleccionadas com base no cálculo da função prioridade)
P_b = ( ) (conjunto de tarefas seleccionadas com base no cálculo dos majorantes)
$\forall j \in S$,
Calcular Pr_j
i = ind max (Pr_j)
$Pr^M = Pr_i$
Seja, $P_a = \{j: j \in S \text{ e } Pr_j \geq (1-p) * Pr^M\}$
$\forall j \in P_a$
Calcular Ub_j,
w = ind min (Ub_j)
$Ub_m = Ub_w$
Seja $P_b = \{j: j \in P_a \text{ e } Ub_j < (1+s) * Ub_m\}$
Escolher aleatoriamente $l \in P_b$

FIGURA 18 - Passo 2.1 dos Algoritmos ACTSALEAT (LIN+EDD_A, LIN+FC_A, LIN+WSWL_A, EXP+EDD_A, EXP+FC_A, EXP+WSWL_A, ACL+EDD_A ACL+FC_A, ACL+WSWL_A)

4 - Experiência Computacional

4.1 - Metodologia de Geração das Instâncias de Teste do Problema

Não existindo no contexto dos problemas de escalonamento um conjunto de instâncias com as quais os diferentes trabalhos de investigação possam, independentemente, testar os algoritmos desenvolvidos, foi seguida na geração das instâncias-teste uma metodologia que decorre da utilizada por Abdul-Razaq e Potts (1988), Ow e Morton (1989) e Sousa e Wolsey (1992). Foram geradas aleatoriamente um total de 84 instâncias para o problema *MCTDM*, com 10, 20 e 50 tarefas.

Numa qualquer instância do *MCTDM*, uma tarefa j é definida por um conjunto de quatro parâmetros: tempos de processamento (p_j), penalidade de antecipação (a_j), penalidade de atraso (b_j) e *due date* (d_j).

Os tempos de processamento de cada tarefa j foram gerados a partir de uma distribuição uniforme no intervalo dos inteiros $\{1, \ldots, p_{max}\}$.

A partir desses valores calculou-se para cada instância o horizonte temporal de planeamento $M = \sum_{j \in N} p_j$.

Para as penalidades por não cumprimento das *due dates* considerou-se a hipótese de dependerem dos tempos de processamento. Desta forma pretendeu-se ter em conta que, quanto maior for o tempo de processamento de uma tarefa, mais importante tenderá a ser o cumprimento da sua *due date*, pois maior será o seu valor

incorporado (aproximado pelo seu conteúdo em trabalho). Deste modo, para penalidades por unidade de tempo de processamento A_j ou B_j iguais, terá maior importância o cumprimento da *due date* da tarefa com maior tempo de processamento.

As penalidades por unidade de tempo de processamento, $A_j = (a/p)_j$ e $B_j = (b/p)_j$, foram geradas aleatoriamente obtendo-se em seguida a_j e b_j através da multiplicação de A_j e B_j por p_j. A_j e B_j foram gerados através de uma distribuição uniforme no intervalo dos inteiros $\{1,\ldots,C_{max}\}$.

As características das diferentes instâncias do problema são também controladas através dos parâmetros que definem a distribuição das *due dates*.

Consideram-se dois parâmetros de controlo da distribuição das *due dates*, τ e ρ, respectivamente o factor de atraso (proporção de tarefas que se espera estejam atrasadas numa sequência aleatoriamente definida) e de dispersão das *due dates*.

As *due dates* foram geradas uniformemente no intervalo dos inteiros $\{ M * (1 - \tau - \frac{\rho}{2}) , M * (1 - \tau + \frac{\rho}{2}) \}$.

Com o objectivo de considerar diferentes combinações de τ e ρ, fez-se variar os parâmetros no conjunto {0.2, 0.4, 0.6, 0.8, 1.0}, calculando-se $Dl = 1 - \tau - \frac{\rho}{2}$ e $Du = 1 - \tau + \frac{\rho}{2}$ conforme Tabela 2:

TAU	RHO	Dl	Du
0.2	0.2	0.7	0.9
0.2	0.4	0.6	1.0
0.2	0.6	0.5	1.1
0.2	0.8	0.4	1.2
0.2	1.0	0.3	1.3
0.4	0.2	0.5	0.7
0.4	0.4	0.4	0.8
0.4	0.6	0.3	0.9
0.4	0.8	0.2	1.0
0.4	1.0	0.1	1.1
0.6	0.2	0.3	0.5
0.6	0.4	0.2	0.6
0.6	0.6	0.1	0.7
0.6	0.8	0.0	0.8
0.6	1.0	-0.1	0.9
0.8	0.2	0.1	0.3
0.8	0.4	0.0	0.4
0.8	0.6	-0.1	0.5
0.8	0.8	-0.2	0.6
0.8	1.0	-0.3	0.7
1.0	0.2	-0.1	0.1
1.0	0.4	-0.2	0.2
1.0	0.6	-0.3	0.3
1.0	0.8	-0.4	0.4
1.0	1.0	-0.5	0.5

TABELA 2 - Valores de Dl e Du

Foram considerados os pares *Dl* e *Du* em que *Dl* é estritamente positivo. As diferentes características das instâncias correspondentes a diferentes combinações de valores para os parâmetros τ e ρ, poderão influenciar de forma diferente o desempenho dos vários algoritmos testados.

As instâncias-teste construídas serão reconhecidas por uma ou duas letras de numeração romana, indicando o número de tarefas, e por um ou dois algarismos, indicando o número de ordem dentro das instâncias-teste com o mesmo número de tarefas.

Os valores de C_{max} e p_{max} para cada uma das instâncias-teste geradas encontram-se na Tabela 3:

	C_{max}	p_{max}
Xq, q = 1,...,14	5	10
Xq, q = 15,...,28	5	100
XXq, q = 1,...,14	5	10
XXq, q = 15,...,28	5	100
Lq, q = 1,...,14	5	10
Lq, q = 15,...,28	5	100

TABELA 3 - Valores de C_{max} e p_{max}

Cada conjunto de valores de p_j, a_j, e b_j é comum às 14 diferentes

instâncias do problema que resultam de utilizar os diferentes pares (*Dl*, *Du*) considerados.

Para cada instância-teste são os seguintes os valores de τ e ρ :

						TAU	RHO
X1	X15	XX1	XX15	L1	L15	0.2	0.2
X2	X16	XX2	XX16	L2	L16	0.2	0.4
X3	X17	XX3	XX17	L3	L17	0.2	0.6
X4	X18	XX4	XX18	L4	L18	0.2	0.8
X5	X19	XX5	XX19	L5	L19	0.2	1.0
X6	X20	XX6	XX20	L6	L20	0.4	0.2
X7	X21	XX7	XX21	L7	L21	0.4	0.4
X8	X22	XX8	XX22	L8	L22	0.4	0.6
X9	X23	XX9	XX23	L9	L23	0.4	0.8
X10	X24	XX10	XX24	L10	L24	0.4	1.0
X11	X25	XX11	XX25	L11	L25	0.6	0.2
X12	X26	XX12	XX26	L12	L26	0.6	0.4
X13	X27	XX13	XX27	L13	L27	0.6	0.6
X14	X28	XX14	XX28	L14	L28	0.8	0.2

Tabela 4 - Valores de Tau e Rho para cada instância-teste gerada

4.2 - Medidas de Avaliação dos Procedimentos Heurísticos

Na avaliação dos algoritmos implementados utilizaram-se algumas das medidas apresentadas em David Dannenbring (1977).

Não se dispondo do valor da solução óptima das diferentes instâncias nem de minorantes para esse valor, a opção seguida foi a de tomar como termo de comparação o valor da melhor solução conhecida.

As medidas de avaliação consideradas foram:

- Erro Relativo (medido, em percentagem, a partir do valor da melhor solução conhecida): ER_{min}

$$ER_{min} = 100 * \left[\left(\frac{MS}{MS_{st}}\right) - 1\right] ,$$

Em que MS é o valor da solução dada pela heurística e MS_{st} é o valor da melhor solução encontrada.

- Percentagem de soluções que têm valor indêntico ao da melhor solução conhecida: P_{h*}

$$P_{h*} = 100 * \left(\frac{V}{R}\right) .$$

Em que V é o número de vezes que o algoritmo iguala a melhor solução em R instâncias.

4.3 - Resultados Computacionais

São apresentados nesta secção os resultados computacionais obtidos com a aplicação das heurísticas descritas no Capítulo 3.

Os algoritmos testados foram codificados em Pascal, tendo sido utilizado um compilador TURBO 5.0. Os tempos de computação (medidos em tempo de C.P.U.) são indicados em segundos e foram obtidos num computador IBM Compatível com processador 80386, no sistema operativo MS/DOS 5[5].

4.3.1 - Regras Simples de Escalonamento

As regras simples de escalonamento implementadas revelam um comportamento distinto para diferentes dimensões do problema.

As três regras são bastantes rápidas, com tempos de computação idênticos e inferiores, em média, a um terço de segundo. Deste ponto de vista há uma ligeira vantagem para a regra *WSWL*.

A qualidade média das soluções é, como seria de esperar, relativamente pobre. Contudo com a regra *WSWL* consegue-se obter, para problemas de 10 tarefas, uma solução de valor idêntico ao da melhor conhecida, em 43% dos casos, com erros médios da ordem dos 5%. Apesar dos seus resultados piorarem bastante com o aumento do número de tarefas, para os problemas com 20 tarefas consegue-se ainda obter uma solução de melhor valor em 11% dos

[5] Os tempos computacionais apresentados incluem a leitura dos dados das instâncias-teste.

casos (ver Tabela 5 e A2 em anexo).

A regra *EDD* demonstra um comportamento semelhante quanto ao aumento dos ER_{min} com a dimensão das instâncias, mas nunca conseguindo obter soluções de valor igual ao melhor.

Por seu lado a heurística *greedy*, *FC*, tem um comportamento mais irregular.

	N=10	N=20	N=50
EDD	10.298	25.864	48.475
WSWL	5.099	34.507	110.241
FC	20.373	46.011	35.675

Tabela 5 - ERmin médios das regras simples de escalonamento

	N=10	N=20	N=50
EDD	0.277	0.280	0.295
WSWL	0.224	0.228	0.239
FC	0.278	0.287	0.323

Tabela 6 - Tempos médios de computação das regras simples de escalonamento

A Tabela 7 resume o comportamento das melhores regras para cada par de valores de τ e ρ [6] (construído a partir dos dados da Tabela A3, em anexo).

RHO / TAU	Baixo	Médio	Alto
Baixo	WSWL	FC	FC
Médio	WSWL	EDD	EDD/FC
Alto	WSWL	-	-

Tabela 7 - Regras Simples de Escalonamento Mais Competitivas Segundo os Valores de Tau e Rho

A regra *WSWL* tem um melhor comportamento em instâncias com menor dispersão das *due dates*, o que é natural já que constitui a

[6] Tau e rho são considerados **baixos** quando são iguais a 0.2, **médios** quando assumem os valores 0.4 e 0.6 e **altos** para 0.8 e 1.0.

situação mais próxima da *due date* única, na base da qual é deduzida a Propriedade III, que está na origem desta regra de escalonamento.

A regra *EDD* tem melhores resultados quando a dispersão das *due dates* é maior e se torna menos conflituosa a sequenciação das tarefas segundo a sua ordem de *due date* (em situações extremas de grande dispersão das *due dates* pode até haver situações de escalonamento sem qualquer conflito entre o cumprimento das *due dates* das diferentes tarefas).

Os resultados do escalonamento obtidos para a instância X28 constituem uma boa ilustração destes fenómenos.

TAREFA	A_j	B_j	D_j	P_j
J1	153	76	156	33
J2	59	19	156	13
J3	17	8	102	6
J4	53	43	92	23
J5	94	115	89	43
J6	418	405	158	89
J7	168	380	82	86
J8	412	439	151	93
J9	293	392	121	87
J10	336	348	142	82

Tabela 8 - Dados da instância X28

As sequências obtidas com a aplicação de cada uma destas regras à resolução desta instância constam da Tabela seguinte:

POSIÇÃO	1	2	3	4	5	6	7	8	9	10	Custo
WSWL	J7	J8	J6	J9	J10	J5	J1	J4	J2	J3	355030
FC	J7	J3	J4	J1	J2	J5	J10	J6	J9	J8	466248
EDD	J7	J5	J4	J3	J9	J10	J8	J1	J2	J6	429474

Tabela 9 - Escalonamentos das tarefas da instância X28 através das regras simples de escalonamento

Neste exemplo de escalonamento são revelados alguns dos erros tipicamente cometidos com as regras simples de escalonamento.

A regra *FC* tende a considerar de uma forma errada as penalidades envolvidas pois, por exemplo, as tarefas *J1* e *J3* são escalonadas muito precocemente, ao contrário da tarefa *J8* que por ter penalidades de atraso muito elevadas é "empurrada" para o final

da sequência com uma penalização muito elevada.

A regra *EDD* revela nesta instância o seu mau comportamento em situações de fraca dispersão das *due dates*, comportamento que tende a ser agravado com o aumento do número de tarefas atrasadas, o que está ligado ao facto de com a regra *EDD* não se tomar em consideração as penalidades associadas a cada tarefa.

4.3.2 - Regras de Função Prioridade

Foi feito um estudo de afinação do parâmetro k das especificações linear e exponencial de Ow e Morton.

Os resultados obtidos com o escalonamento revelaram ser bastante sensíveis à definição deste parâmetro, degradando-se bastante com a especificação de um valor para k não ajustado, especialmente na versão exponencial.

Em anexo, Gráficos A1 a A6, são apresentados os valores dos Er_{min} obtidos para as diferentes dimensões das instâncias em N e P_{max}.

O parâmetro θ foi fixado no valor dois para todas as implementações realizadas, podendo merecer alguns afinamentos para diferentes dimensões das instâncias dos problemas mas que não se revelaram tão importantes quanto os de k.

Em média, os resultados obtidos com a função *LIN-ET* são melhores do que para as restantes com N=10 e N=20, mas com N=50 é com a função exponencial (*EXP-ET*) que se obtêm os melhores resultados (ver Tabela 10).

ER_{min}	N=10	N=20	N=50
LIN-ET	0.315	3.214	6.121
EXP-ET	0.454	3.296	5.419
ACL	1.422	5.079	10.618

Tabela 10 - ER_{min} para algoritmos com função prioridade

Se tomarmos em consideração a percentagem de instâncias em que se obtem a melhor solução com cada uma das três regras heurísticas, estas conclusões são confirmadas.

Não é possível concluir sobre um melhor comportamento de uma função sobre as restantes, pois com nenhuma delas se obtem sistematicamente os melhores valores (ver Tabela 11) apesar de se verificarem melhores resultados médios com as versões *LIN-ET* e *EXP-ET*.

P_h^*	N=10	N=20	N=50
LIN-ET	85.741%	57.143%	53.571%
EXP-ET	71.429%	53.571%	32.143%
ACL	28.571%	25.000%	21.429%

Tabela 11 - Percentagem de instâncias em que cada uma das regras obtem a melhor solução de entre as três funções de prioridade

Estes resultados indiciam que existe um comportamento distinto entre as regras para diferentes instâncias.

Se se analisarem os resultados obtidos segundo os factores que controlam a distribuição das *due dates* conseguem-se intuir alguns padrões de comportamento entre as distintas funções (ver Tabela A3).

A função linear tende a ter melhor comportamento para situações em que a dispersão das *due dates* é menor, enquanto que a versão exponencial tende a superar as outras duas em instâncias em que a dispersão das due dates não seja máxima nem mínima.

Quando a dispersão das *due dates* é alta e o factor de atraso é baixo a função *ACL* tende a produzir resultados competitivos com os das outras funções (instâncias 4, 18, 5 e 19, ver Tabela A3).

Quando o factor de atraso aumenta os seus resultados relativos

demonstram alguma deterioração.

Este resultado confirma a ideia de que a visão "optimista" da função *ACL*, face a situações em que há uma grande proporção de tarefas atrasadas, não proporciona um bom mecanismo de discriminação.

Os tempos médios de computação são sensivelmente o dobro dos anteriores mas ainda, em média, abaixo de um segundo (ver Tabela 12).

TMC	N=10	N=20	N=50
LIN-ET	0.556	0.567	0.584
EXP-ET	0.612	0.619	0.652
ACL	0.609	0.620	0.648

Tabela 12 - Tempo Médio de Computação (TMC) para os algoritmos com regras de função prioridade

4.3.3 - Regras com Antecipação do Custo Total da Sequência

Na implementação feita dos algoritmos com antecipação do custo total da sequência, foram calculados majorantes para o custo das sequências resultantes de sequenciar na iteração corrente, todas as tarefas cuja prioridade, avaliada pelas respectivas funções, não fosse mais do que 15% inferior à prioridade máxima (p = 0.15).

A combinação das regras simples com as funções de prioridade, permite obter resultados que, medidos em termos de ER_{min}, são, em média, melhores para problemas com maior número de tarefas (ver Tabela 13). Esta situação poderá estar associada aos baixos valores dos ER_{min} obtidos nos problemas com menor número de tarefas com as regras com funções de prioridade. No entanto, para problemas de dimensão média os resultados poderiam eventualmente ser melhorados se se considerassem valores de p mais adequados a esses problemas.

ER_{min}	N=10	N=20	N=50
LIN + EDD	0.809	3.122	4.677
EXP + EDD	1.681	4.622	4.964
ACL + EDD	1.861	4.724	7.506
LIN + FC	1.454	3.617	5.223
EXP + FC	2.310	4.733	5.668
ACL + FC	1.619	4.995	7.959
LIN + WSWL	0.628	3.358	5.042
EXP + WSWL	1.454	3.617	7.056
ACL + WSWL	1.915	5.377	8.351

Tabela 13 - ER_{min} para os algoritmos com antecipação do custo total da sequência

Os tempos de computação médios não cresceram de forma assinalável, excepto para alguns algoritmos e em problemas de 50 tarefas.

TMC	N=10	N=20	N=50
LIN + EDD	0.603	0.681	1.179
EXP + EDD	0.691	1.491	6.461
ACL + EDD	0.684	0.994	2.453
LIN + FC	0.619	0.750	1.353
EXP + FC	0.690	1.501	7.308
ACL + FC	0.736	0.990	2.676
LIN + WSWL	0.600	0.680	1.159
EXP + WSWL	0.687	1.480	5.817
ACL + WSWL	0.680	0.899	2.026

Tabela 14 - TMC para os algoritmos com antecipação do custo total da sequência

Para os problemas de dimensão mais reduzida (N=10 e N=20) as combinações de regras com que se obtêm os melhores resultados (avaliados pelo ER_{min}) são as que resultam de conjugar as regras que isoladamente deram melhores soluções (ver Tabela 15).

Combinações das melhores regras		
N=10	WSWL + LIN-ET	
N=20	EDD + LIN-ET	
N=50	FC + EXP-ET	
Melhores combinações		
	Da regra simples	Da função prioridade
N=10	WSWL + LIN-ET	WSWL + LIN-ET
N=20	EDD + LIN-ET	EDD + LIN-ET
N=50	FC + LIN-ET	EDD + EXP-ET

Tabela 15 - Combinações das melhores regras e melhores combinações

4.3.4 - Pesquisa Local

Os resultados obtidos com os diferentes algoritmos de pesquisa local implementados estão resumidos na Tabela 16.

No geral obtiveram-se bons resultados se os compararmos com os dos algoritmos construtivos determinísticos, em especial quando a solução inicial é obtida pela regra de função linear (*LIN-ET*).

Solução Inicial	Dimensão	Tr	ERmin	TMC
LIN-ET	N=10	0	0.181	0.558
	N=20	3	1.138	0.601
	N=50	13	3.593	0.657
WSWL	N=10	2	1.117	0.322
	N=20	7	16.475	0.332
	N=50	59	76.124	0.399
EDD	N=10	5	1.174	0.329
	N=20	27	4.867	0.384
	N=50	236	18.524	0.418
ACL	N=10	2	0.675	0.718
	N=20	6	4.194	0.783
	N=50	13	10.306	0.812

Tabela 16 - Resultados dos Algoritmos de Pesquisa Local

Nota: Tr - Número médio de trocas efectuadas
TMC - Tempo médio de computação

O número médio de trocas realizadas quando a solução de partida é construída a partir de algum dos algoritmos que utilizam os resultados da condição de adjacência (*WSWL*, *LIN-ET* e *ACL*) é extremamente reduzido, o que não contraria as expectativas.

Dada a má qualidade de algumas dessas soluções de partida, nomeadamente as obtidas com recurso à regra *WSWL*, para a obtenção de soluções finais de boa qualidade seria necessária a realização de um número de trocas bastante superior que permitissem transformar essa solução, o que não é conseguido com esta implementação. Note-se que em média, para problemas com 50 tarefas, os ER_{min} finais são de 76%, apenas se conseguindo melhorias sobre a solução inicial da ordem dos 14%.

Situação de tipo contrário é a que acontece quando a solução inicial é obtida com a regra *EDD*, em que o algoritmo de pesquisa local consegue operar maiores transformações sobre essa solução obtendo-se óptimos locais mais competitivos, especialmente com problemas de dimensão média, para os quais se conseguem soluções finais com ER_{min} de 5% e com melhorias de 15% sobre as soluções iniciais.

A qualidade das soluções finais depende de uma forma significativa da qualidade das soluções iniciais. Quando estas são de má qualidade, chegam-se a resultados finais também de má qualidade. Este comportamento leva a concluir que a vizinhança utilizada é do tipo fraco.

4.3.5 - Simulated Annealing

A implementação de algoritmos do tipo *S.A.* requer a definição de um conjunto de parâmetros de inicialização que estão ligados às questões genéricas do processo de *annealing* referidas na Secção 3.3.1.

Não se conhecem referências a trabalhos efectuados na aplicação do *S.A.* a problemas de escalonamento do tipo do *MCTDM*. Van Laarhoven, Aarts e Lenstra (1992) descrevem uma aplicação a um problema de *Job Shop*, citando apenas uma outra aplicação ao mesmo problema.

Na aplicação feita neste trabalho foram tidas em consideração algumas das conclusões da aplicação do *S.A.* a outros problemas de optimização combinatória, embora com a consciência de que elas não são totalmente generalizáveis. Citam-se, em particular, os trabalhos de Johnson et al (1989 e 1991) e Alves, M.L. (1991).

O estudo feito sobre os parâmetros não é exaustivo, tendo a atenção recaído sobre os valores a considerar para a temperatura inicial ($TEMP_0$), parâmetro fundamental num processo de *annealing* por estar associado ao Quociente de Aceitação (*Q.A.*) das transições não melhorativas no seu estado inicial.

O conjunto de parâmetros para os quais é necessário definir valores (ver Figura 16) são: $TEMP_0$ (temperatura inicial); *SIZEFACTOR* (multiplicador do número de transições realizadas a cada tamperatura); *CUTOFF* (multiplicador do número de transições aceites à mesma temperatura); *TEMPFACTOR* (factor de redução da temperatura); *NRU* (número de corridas de "annealing" realizadas); *FREEZELIM* (critério do estado de congelação); *MINPERCENT* (percentagem de transições aceites a uma dada temperatura a baixo da qual se realiza uma aproximação ao estado de congelação).

Para a determinação da temperatura inicial e tendo em conta as conclusões dos estudos referidos, considerou-se:

$$\chi(TEMP) = \frac{número\ de\ transições\ propostas}{número\ de\ transições\ aceites}$$

o *Q.A.* das transições propostas a uma dada temperatura e que se deveria escolher uma temperatura de inicialização que permitisse $X(TEMP_0)$ aproximadamente igual a 0.4.

A Tabela 17 resume os resultados obtidos:

N=10 PMAX=10	$TEMP_0$ Q.A.	15 0.574	10 0.509	7.5 0.461	5 0.392
N=10 PMAX=100	$TEMP_0$ Q.A.	130 0.325	145 0.346	160 0.357	190 0.388
N=20 PMAX=10	$TEMP_0$ Q.A.	30 0.775	15 0.681	10 0.614	7.5 0.556
N=20 PMAX=100	$TEMP_0$ Q.A.	260 0.629	130 0.487	120 0.469	100 0.433
N=50 PMAX=10	$TEMP_0$ Q.A.	40 0.797	20 0.717	10 0.638	5 0.550
N=50 PMAX=100	$TEMP_0$ Q.A.	580 0.648	260 0.533	200 0.492	150 0.449

Tabela 17 - Quocientes de Aceitação (Q.A.) à Temperatura Inicial

Para *SIZEFACTOR*, atendendo à reduzida dimensão das vizinhanças consideradas, *VIZ* = 2 * (N - 1), foi considerado o valor de 16.

Não tendo sido feito um estudo exaustivo da interferência de cada um dos parâmetros na evolução do *annealing* foi considerado para *CUTOFF* um valor idêntico ao de *SIZEFACTOR*, eliminando-se deste modo um dos critérios de limitação do número de transições a propor a cada temperatura.

Para o factor de redução da temperatura foram testados valores entre 0.75 e 0.99.

Atendendo aos valores iniciais da temperatura e a que as soluções aceites se deterioram bastante nas primeiras transições fazendo convergir rapidamente o processo de *annealing* para o estado de congelação, foi considerado para as corridas realizadas um *TEMPFACTOR* igual a 0.95.

Este valor surge como uma solução de compromisso entre uma convergência muito lenta e a temperaturas muito elevadas, com *TEMPFACTOR* superior a 0.95 e uma convergência muito rápida com valores inferiores.

O número de corridas realizadas não pareceu ser muito importante

O número de corridas realizadas não pareceu ser muito importante na qualidade dos resultados obtidos, pelo que, por motivos de redução do tempo de computação, foram realizadas dez corridas para os problemas de 10 e 20 tarefas e cinco para os problemas de 50 tarefas.

Os valores de *FREEZELIM* e *MINPERCENT* condicionam a rapidez com que se atinge o estado de congelação.

A análise do custo médio das soluções aceites ao longo de uma corrida de *annealing*, de que se ilustram duas situações nos Gráficos 2 e 3, revela o comportamento desta aplicação do *S.A.* ao *MCTDM*.

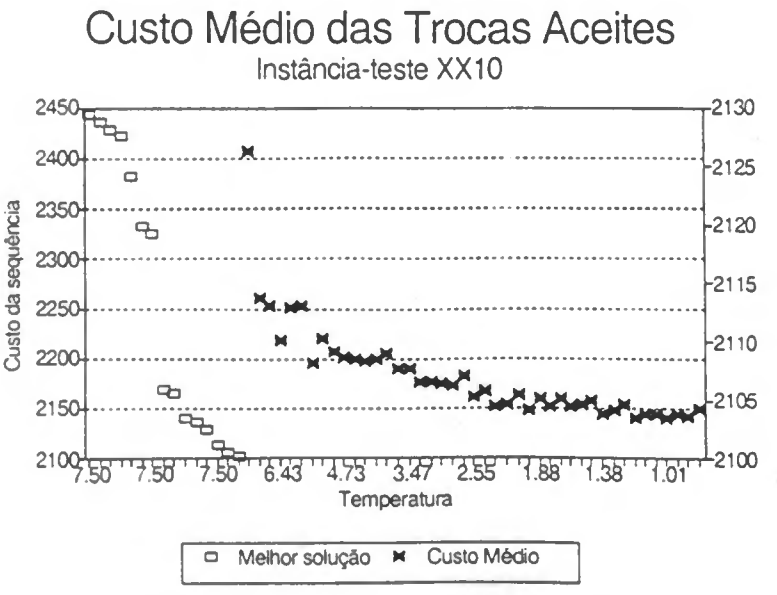


Gráfico 2 - Evolução das melhores soluções conhecidas e custo médio das trocas aceites durante uma corrida da algoritmo Simulated Annealing para a instância XX10[7]

[7] No eixo vertical a escala do lado esquerdo refere-se às melhores soluções encontradas e a do lado direito ao custo médio das transições efectuadas a uma mesma temperatura.

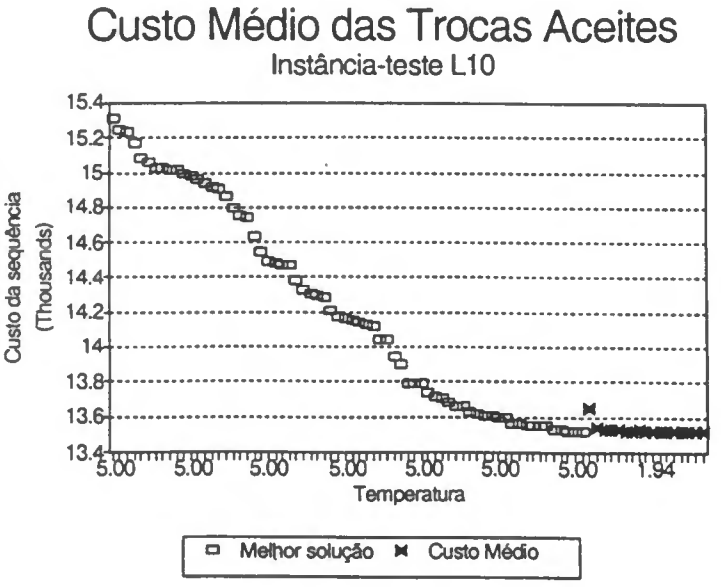


Gráfico 3 - Evolução das melhores soluções conhecidas e custo médio das trocas aceites durante uma corrida da algoritmo Simulated Annealing para a instância L10

Nestes gráficos constata-se que, as trocas que levam à obtenção de soluções melhores do que as conhecidas até então, decorrem, quase exclusivamente, à temperatura inicial.

A partir do momento em que a temperatura começa a decrescer o custo das soluções aceites evolui de forma controlada, isto é, não se deteriora muito, mas não se consegue, no geral, obter melhores soluções.

Em corridas em que a temperatura inicial foi fixada a um nível superior do *Q.A.*, verificaram-se comportamentos semelhantes, mas o custo médio das soluções aceites às temperaturas mais baixas era superior.

Desta forma os valores de *FREEZELIM* e de *MINPERCENT* apenas podem fazer prolongar este processo, aparentemente sem vantagens em termos da qualidade da solução final. *FREEZELIM* foi fixado em 3 para N=10, 2 para N=20 e 1 para N=50 e *MINPERCENT* é de 2% para todas as instâncias. Assim admite-se uma aproximação ao estado de congelação apenas em situações de muito baixo número de

transições aceites.

Todas estas considerações estão resumidas na Tabela 18.

Instâncias / Parâmetros	N=10 PMAX=10	N=10 PMAX=100	N=20 PMAX=10	N=20 PMAX=100	N=50 PMAX=10	N=50 PMAX=100
TEMPO	7.5	190	7.5	100	5	150
SIZEFACTOR	16	16	16	16	16	16
CUTOFF	16	16	16	16	16	16
TEMPFACTOR	0.95	0.95	0.95	0.95	0.95	0.95
NRU	10	10	10	10	5	5
FREEZELIM	3	3	2	2	1	1
MINPERCENT	0.02	0.02	0.02	0.02	0.02	0.02

Tabela 18 - Valores dos parâmetros do Algoritmo Simulated Annealing

Para completar a descrição das características da implementação computacional do algoritmo é de referir ainda que a solução inicial foi construída através da regra heurística *LIN-ET*. Esta escolha foi motivada pelo facto de ter sido sobre esta solução que a pesquisa local se mostrou mais forte.

	N=10 ER_{min}	N=10 TMC	N=20 ER_{min}	N=20 TMC	N=50 ER_{min}	N=50 TMC
ANNEALING	0.181	15.179	1.138	45.879	2.699	138.664

Tabela 19 - Resultados do Algoritmo Simulated Annealing

Os resultados obtidos com o *S.A.* são apresentados na Tabela 19.

Em média os tempos de computação obtidos com o *S.A.* são extremamente elevados se comparados com os dos restantes algoritmos implementados.

De uma forma geral, estes tempos de computação não são compensados com a obtenção de soluções com uma qualidade de proporcionalidade semelhante, em especial para problemas de 10 e 20 tarefas. Nestes casos obtêm-se ER_{min} médios idênticos aos da pesquisa local, quando os dois algoritmos partem da mesma solução inicial e o *S.A.* tem *TMC* extremamente superiores.

Para problemas de 50 tarefas os valores de ER_{min} são mais

reduzidos, mas os tempos de computação começam a tornar-se demasiado grandes quando comparados com os obtidos para os outros algoritmos.

4.3.6 - Algoritmos Construtivos Aleatorizados

A aleatorização dos algoritmos construtivos que recorrem apenas ao cálculo da função prioridade necessitam da fixação do parâmetro *p*, que representa a percentagem máxima que a prioridade de uma tarefa pode distar da prioridade máxima, para que possa ser elegível para escalonamento e o número de corridas (*NRUNS*) a realizar do algoritmo. Na implementação feita testou-se para *NRUNS* os valores 10 e 20 e *p* foi fixado em 0.15.

A aleatorização dos algoritmos com antecipação do custo total das sequências necessita também da fixação do parâmetro *s*, percentagem máxima que os majorantes calculados para o valor de cada sequência se podem afastar do menor majorante. Para estes algoritmos foram testados valores para *NRUNS* igual a 10 e a 20, *p* igual a 0.15 e *s* igual a 0.2.

A Tabela 20 apresenta os resultados obtidos com as heurísticas construtivas em que são tomadas decisões de forma aleatorizada.

	N=10		N=20		N=50	
	ER_{max}	TMC	ER_{max}	TMC	ER_{max}	TMC
LINET_A	0.235	0.698	2.075	0.875	2.690	1.812
EXPET_A	0.460	0.690	2.941	0.942	3.282	2.282
ACL_A	1.074	0.701	3.607	1.159	6.257	4.616
LIN+EDD_A	0.190	0.766	2.569	1.999	2.466	14.060
LIN+FC_A	0.200	1.075	2.189	2.158	2.853	15.887
LIN+WSWL_A			2.539	3.042	2.450	19.522
EXP+EDD_A	0.461	0.795				

Tabela 20 - Resultados dos Algoritmos Construtivos com Aleatorização[8]

No seu conjunto os resultados obtidos foram bastante satisfatórios.

[8]São apresentados os resultados após 20 corridas dos algoritmos e das heurísticas compostas apenas se apresentam os resultados dos três melhores algoritmos.

Em termos da relação tempo de computação/custo das soluções estes algoritmos apresentam valores bastante competitivos em relação oa algoritmo *S.A.* implementado.

Com o crescimento do número da tarefas estes algoritmos tendem a obter resultados mais competitivos, atingindo-se valores de Er_{min} da ordem dos 2.5% em problemas de 50 tarefas para mais do que um algoritmo.

Estes valores são tanto mais significativos se os analisarmos juntamente com os da Tabela A2. Este cruzamento permite obter uma ideia da discrepância dos resultados ao longo das 28 instâncias do problema para cada dimensão considerada.

Os algoritmos construtivos aleatorizados tendem a obter resultados mais competitivos quando medidos em termos de ER_{min} do que em termos de P_h^* o que significa que a magnitude dos erros em relação à melhor solução conhecida é menor do que a dos outros algoritmos.

Com tempos de computação controlados conseguem-se obter resultados, medidos pelos ER_{min}, que para alguns desses algoritmos são comparáveis aos dos *S.A.*.

Na Tabela 21 são apresentados os valores para a percentagem de melhoria entre a melhor solução obtida ao fim de 10 corridas e a obtida ao fim de 20 corridas.

Regra Heurística	N=10	N=20	N=50
LINET_A	0.069%	0.270%	0.252%
EXPET_A	0.097%	0.168%	0.439%
ACL_A	0.089%	0.182%	0.456%
LIN+EDD_A	0.076%	0.234%	0.400%
EXP+EDD_A	0.056%	0.101%	0.373%
ACL+EDD_A	0.214%	0.328%	0.614%
LIN+FC_A	0.043%	0.132%	0.344%
EXP+FC_A	0.086%	0.334%	0.465%
ACL+FC_A	0.057%	0.171%	0.572%
LIN+WSWL_A	0.056%	0.124%	0.376%
EXP+WSWL_A	0.175%	0.180%	0.634%
ACL+WSWL_A	0.091%	0.195%	0.502%

Tabela 21 - Melhoria do valor das soluções óptimas obtidas entre 10 e 20 corridas dos algoritmos

Ao fim de 10 corridas, os melhores resultados obtidos não são, percentualmente, muito distintos dos obtidos ao fim de 20 corridas. Em média, para nenhum dos algoritmos e dimensões do problema obtiveram melhorias superiores a 0.65%.

Como a diferença entre os ER_{min} dos algoritmos mais competitivos é sempre na ordem das décimas, a realização de corridas de duração equivalente às do *S.A.* pode permitir obter soluções de qualidade semelhante ou superior às por este obtidas, em especial para os algoritmos com regras de função prioridade.

5 - Conclusão

Nesta dissertação é estudado um problema de escalonamento de tarefas numa máquina em que são impostas penalidades pelo atraso ou antecipação da conclusão de cada tarefa em relação à data pretendida para o seu término, *due date*, que se supõe exógena. Admite-se ainda que diferentes tarefas tenham diferentes *due dates*.

O objectivo a optimizar consiste na soma dos custos de desvio de cada tarefa (Minimização dos Custos Totais de Desvio - *MCTDM*).

O estudo de problemas de escalonamento de tarefas em que se consideram simultaneamente penalidades de antecipação e atraso é uma área de investigação que apenas recentemente tem vindo a ganhar um interesse crescente. Este facto não deverá estar desligado da também crescente importância que os custos de antecipação assumiram nos modelos de gestão industrial do tipo *Just-in-Time* e que nalgumas das suas componentes é formalizável por problemas como o *MCTDM*.

Dado que a utilização de penalidades de antecipação e atraso dá lugar a funções objectivo não regulares, em que a penalidade total pode decrescer com o crescimento dos momentos de conclusão das tarefas, têm surgido associados a estes problemas um conjunto de novos procedimentos, para obtenção de soluções, distintos dos aplicados aos problemas com funções objectivo regulares.

No geral os problemas de escalonamento com função objectivo linear não regular, são *NP-Difíceis* desde que as *due dates* sejam exógenas e pelo menos uma distinta das restantes (Garey, Tarjan e Wilfong, 1988).

Apesar desta característica geral, para alguns casos particulares do *MCTDM* são conhecidas propriedades das soluções óptimas e para outros são mesmo conhecidos algoritmos capazes de os resolver em tempo polinomial.

Infelizmente, para o *MCTDM* não são conhecidas quaisquer propriedades das soluções óptimas que possam permitir a sua resolução de forma mais eficiente utilizando algoritmos de enumeração implícita do tipo *branch-and-bound* e *programação dinâmica*.

Como é feito para outros problemas de optimização combinatória com estas características extremas de dificuldade desenvolvem-se nesta dissertação alguns métodos heurísticos cujos resultados são comparados com os de alguns algoritmos similares existentes na literatura.

Os algoritmos apresentados dividem-se em duas classes, os de tipo construtivo e os de tipo melhorativo. Para estas duas classes foram implementadas versões determinísticas e versões aleatorizadas.

No conjunto dos algoritmos construtivos desenvolveu-se uma nova regra simples de escalonamento, construída a partir do resultado de uma condição de adjacência das tarefas numa sequência óptima, que quando comparada com regras semelhantes existentes na literatura apresenta bons resultados em problemas de pequena dimensão.

Ainda nos algoritmos construtivos determinísticos foi desenvolvida uma nova especificação para uma função de prioridade que, partindo de uma ideia "optimista" sobre a sequenciação de cada tarefa nas posições corrente e seguinte, selecciona para tarefa a escalonar aquela cuja penalização mais se deteriora se não for escalonada imediatamente. Os resultados obtidos, quando comparados com os de outras funções de prioridade existentes na literatura não são, no geral, melhores, revelando alguma

competitividade em situações em que o factor de atraso é baixo e a dispersão das *due dates* elevada.

Foram ainda implementados algoritmos construtivos determinísticos em que se calcula, em cada iteração, um majorante para o custo das sequências resultantes de escalonar na posição corrente cada tarefa ainda por escalonar. Estes procedimentos constituem uma simplificação dos propostos por Ow e Morton (1989).

No campo dos algoritmos melhorativos foram implementadas versões de algoritmos do tipo pesquisa local (determinístico) e *simulated annealing* (aleatorizado).

A pesquisa local implementada permitiu pôr em evidência que a obtenção de soluções de melhor valor na vizinhança de uma dada solução depende de forma decisiva da estrutura com que essa solução foi construída. As características dessa estrutura dependem das regras de escalonamento utilizadas na sua construção.

Atendendo à relação entre o valor da solução inicial e da melhor solução local encontrada nas vizinhanças utilizadas, é possível concluir que estas são do tipo fraco.

Na implementação feita do algoritmo *simulated annealing* revelaram-se algumas características já reportadas nalgumas implementações deste tipo de algoritmos a outros problemas de optimização combinatória: se no geral se obtêm soluções finais de boa qualidade, não raras vezes isso é conseguido à custa de tempos de computação muitas dezenas de vezes superiores aos de outros algoritmos implementados e com os quais se conseguiram soluções de custo comparável.

Finalmente desenvolveram-se, para alguns dos algoritmos construtivos, versões com aleatorização do processo construtivo. Com tempos de computação controlados conseguiram-se resultados

extremamente competitivos nalgumas das versões desenvolvidas.

Estes algoritmos mostraram-se mais eficazes na pesquisa do espaço das soluções do que os algoritmos melhorativos, permitindo a obtenção de melhorias em relação às versões determinísticas que, no geral, são mais significativas.

A ênfase posta nos procedimentos heurísticos de pesquisa do espaço das soluções admissíveis em problemas *NP-Difíceis* como são a generalidade dos problemas de escalonamento e de que o *MCTDM* é um exemplo, tem aberto a porta à "importação" pela Investigação Operacional de algumas técnicas de pesquisa habituais noutras àreas de investigação, em especial na Inteligência Artificial e noutras ciências da Computação.

Para além das, habitualmente referidas, tentativas de implementação de algoritmos de computação em paralelo (Ow e Morton, 1989), alguma atenção deverá ser dada a novas definições dos espaços de pesquisa, de que se destacam as propostas por Storer, Wu e Vaccari (1992) numa aplicação a um problema de *Job Shop*. Nesta aplicação propõe-se que se transfira a pesquisa do espaço das soluções para dois outros espaços, o dos dados do problema e o das heurísticas de resolução.

Dada a facilidade e generalidade destes procedimentos e a rigidez verificada na estrutura das soluções do *MCTDM*, extremamente dependente dos critérios de escalonamento utilizados na sua construção, a utilização de procedimentos de pesquisa que actuem no espaço das heurísticas de resolução, adaptando a cada fase do escalonamento uma certa heurística "especializada" na resolução de situações particulares de atraso, antecipação e ou congestão de *due dates*, poderá ser um campo de investigação a dedicar alguma atenção no futuro.

6 - Referências Bibliográficas

[1]Abdul-Razaq, T.S. e Potts, C.N. (1988) - *Dynamic Programming State-Space Relaxation for Single-Machine Scheduling*. Journal of Operational Research Society 39, 141-152.

[2]Alves, M.L. (1991) - *Soluções Aproximadas para o Problema de Localização Simples - Algoritmo Simulated Annealing*. I.S.E.G. (U.T.L.) Tese de Mestrado. 1991.

[3]Baker, K., Scudder, G. (1990) - *Sequencing with Earliness and Tardiness Penalties: A Review*. Operations Research 38, 22-36.

[4]Chand, S. e Chhajed, D. (1992) - *A Single Machine Model for Determination of Optimal Due Dates and Sequence*. Operations Research 40, 596-602.

[5]Cheng, T.C.E. (1984) - *Optimal Due-Date Determination and Sequencing of n Jobs on a Single Machine*. Journal of Operational Research Society 35, 433-437.

[6]Cheng, T.C.E. e Gupta, M.C. (1989) - *Survey of Scheduling Research Involving Due-Date Determination Decisions*. European Journal of Operational Research 38, 156-166.

[7]Constantino, M.F. (1989) - *Job-Scheduling com Custos Totais numa Máquina - Minorantes e Algoritmos*. F.C.U.L. Tese de Mestrado. 1989.

[8]Dannenbring, D.G. (1977) - *An Evaluation of Flow Shop Sequencing Heuristics*. Management Science 23, 1174-1182.

[9]Davis, J.S. e Kanet, J.J.a) - ***Single Machine Scheduling with Early and Tardy Completion Costs***. Working Paper, Department of Management, Clemson University, Clemson, SC.

[10]Davis, J.S. e Kanet, J.J.b) - ***Single Machine Scheduling with Non-Regular Convex Completion Costs***. Working Paper, Department of Management, Clemson University, Clemson, SC.

[11]Emmons, H. (1969) - ***One-Machine Sequencing to Minimize Certain Functions of Job Tardiness***. Operations Research 17, 701-715.

[12]Fry, T.D., Armstrong, R.D., Lewis, H. (1989) - ***A Framework for Single Machine Multiple Objective Sequencing Research***. Omega 17, 595-607.

[13]Fry, T.D., Leong, G.K., Rakes, T.R. (1987) - ***Single Machine Scheduling: A Comparison of Two Solution Procedures***. Omega 15, 277-282.

[14]Garey, M.R., Graham, R.L. e Johnson, D.S. (1978) - ***Performance Guarantees for Scheduling Algorithms***. Operations Research 26, 3-21.

[15]Garey, M.R., Tarjan, R.E. and Wilfong, G.T. (1988) - ***One-Processor Scheduling With Symmetric Earliness and Tardiness Penalties***. Mathematics of Operations Research 13, 330-348.

[16]Glover, F. (1989) - ***Tabu Search - Part I***. ORSA Journal on Computing 1, 190-206.

[17]Gupta, S.K., Kyparisis, J. (1987) - ***Single Machine Scheduling Research***. Omega 15, 207-227.

[18]Hall, N., Kubiak, W., Sethi, S. (1991) - ***Earliness-Tardiness Scheduling Problems, II: Deviation of Completion Times About a Restrictive Common Due Date***. Operations Research 39, 847-856.

[19]Hall, N., Posner, M. (1991) - ***Earliness-Tardiness Scheduling Problems, I: Weighted Deviation of Completion Times About a Common Due Date***. Operations Research 39, 836-846.

[20]Hart, J.P., Shogan A.W. (1987) - ***Semi-Greedy Heuristics: An Empirical Study***. Operations Research Letters 6, 107-114.

[21]Heiko, L. (1989) - ***A Simple Framework for Understanding JIT***. Production and Inventory Management Journal, Fourth Quarter, 61-63.

[22]Jonhson, D.S., Aragon, C.R., McGeoch, L.A., Schevon, C. (1989) - ***Optimization by Simulated Annealing: An Experimental Evaluation; Part I, Graph Partitioning***. Operations Research 37, 865-892.

[23]Jonhson, D.S., Aragon, C.R., McGeoch, L.A., Schevon, C. (1991) - ***Optimization by Simulated Annealing: An Experimental Evaluation; Part II, Graph Coloring and Number Partitioning***. Operations Research 39, 378-406.

[24]Kahlbacher, H.G. (1993) - ***Scheduling with Monotonous Earliness and Tardiness Penalties***. European Journal of Operational Research 64, 258-277.

[25]Karmarkar U. (1989) - ***Getting Control of Just-in-Time***. Harvard Business Review September-October 1989, 122-131.

[26]Kirkpatrick, S., Gelatt Jr., C.D. (1983) - ***Optimization by Simulated Annealing***. Science 220, 671-680.

[27]Lakshminarayan, S., Lakshmanan, R., Papineau, R.L., Rochette R. (1978) - ***Optimal Single-machine Scheduling with Earliness and Tardiness Penalties***. Operations Research 26, 1079-1082.

[28]Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., Shmoys, D.B. (1989) - *Sequencing and Scheduling: Algorithms and Complexity*. Report BS-R8909, Centre for Mathematics and Computer Science, Amsterdam.

[29]Malek, M., Guruswamy. M., Owens, H., Pandya, M. (1990) - *A Hybrid Algorithm Technique*. Working Paper. Dept. of Electrical and Computer Engineering. The University of Texas at Austin, 1990.

[30]Ow, P.S., Morton T. (1989) - *The Single Machine Early/Tardy Problem*. Management Science 35, 177-191.

[31]Panwalkar, S.S., Iskander, W. (1977) - *A Survey of Scheduling Rules*. Operations Research 25, 45-61.

[32]Panwalkar, S.S., Rajagopalan, R. (1992) - *Single-machine Sequencing with Controllable Processing Times*. European Journal of Operational Research 59, 298-302.

[33]Panwalkar, S.S., Smith, M.L., Seidman A. (1982) - *Common Due Date Assignment to Minimize Total Penalty for the One Machine Scheduling Problem*. Operations Research 30, 391-399.

[34]Papadimitriou, C.H., Steiglitz, K. (1982) - *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall.

[35]Potts, C.N., Wassenhove L.N.V. (1985) - *A Branch and Bound Algorithm for the Total Weighted Tardiness Problem*. Operations Research 33, 363-377.

[36]Potts, C.N., Wassenhove L.N.V. (1988) - *Algorithms for Scheduling a Single Machine to Minimize the Weighted Number of Late Jobs*. Management Science 34, 843-858.

[37]Quaddus, M.A. (1987) - *A Generalized Model of Optimal Due-Date Assignment by Linear Programming*. Journal of Operational Research Society 38, 353-359.

[38]Rachamadugu, R. M. V., (1987) - *A Note on the Weighted Tardiness Problem*. Operations Research 35, 450-452.

[39]Raghavachari, M. (1986) - *A V-shape Property of Optimal Schedule of Jobs About a Common Due Date*. European Journal of Operational Research 23, 401-402.

[40]Raghavachari, M. (1988) - *Scheduling Problems With Non-Regular Penalty Functions - A Review*. Opsearch 25, 144-164.

[41]Rinnooy Kan, A.H.G., Lageweg, B.J., Lenstra, J.K. (1975) - *Minimizing Total Costs in One-Machine Scheduling*. Operations Research 23, 908-927.

[42]Sen, T., Gupta, S.K. (1984) - *A State-of-Art of Static Scheduling Research Involving Due Dates*. Omega 12, 63-76.

[43]Sousa, J.P., Wolsey, L.A. (1992) - *A Time Indexed Formulation of Non-Preemptive Single Machine Scheduling Problems*. Mathematical Programming 54, 353-367.

[44]Storer, R.H., Wu S.D., Vaccari R. (1992) - *New Search Spaces for Sequencing Problems with Application to Job Shop Scheduling*. Management Science 38, 1495-1509.

[45]Van Laarhoven, P.J.M., Aarts, E.H.L., Lenstra, J.K. (1992) - *Job Shop Scheduling by Simulated Annealing*. Operations Research 40, 113-125.

[46]Zanakis, S.H., Evans, J.R. (1981) - *Heuristic "Optimization": Why, When and How to Use It*. Interfaces 11, 84-90.

ANEXOS

DADOS COMPLEMENTARES

X1	2038	X15	251287	XX1	7470
X2	2110	X16	273587	XX2	6655
X3	2571	X17	319908	XX3	5944
X4	3001	X18	370384	XX4	5220
X5	3491	X19	420961	XX5	4820
X6	1450	X20	190396	XX6	5277
X7	1011	X21	156991	XX7	4514
X8	806	X22	150880	XX8	3655
X9	807	X23	159166	XX9	2827
X10	1406	X24	198628	XX10	2008
X11	2005	X25	209629	XX11	5841
X12	1494	X26	167775	XX12	5905
X13	1386	X27	126082	XX13	5539
X14	3707	X28	355030	XX14	9798
XX15	584423	L1	67235	L15	3806773
XX16	610308	L2	61199	L16	3514391
XX17	683795	L3	58293	L17	3446586
XX18	758807	L4	57598	L18	3193807
XX19	834492	L5	57155	L19	3613300
XX20	323406	L6	42132	L20	2405282
XX21	263050	L7	31594	L21	1933514
XX22	209118	L8	20965	L22	1301614
XX23	189657	L9	13601	L23	1008578
XX24	256111	L10	13083	L24	1135880
XX25	339629	L11	39003	L25	2383228
XX26	286632	L12	30536	L26	1734398
XX27	208487	L13	23125	L27	1178189
XX28	533055	L14	58459	L28	3287349

Tabela A1 - Valores Mínimos Obtidos para cada Instância-Teste Gerada

	N=10	N=20	N=50
EDD	0.000%	0.000%	0.000%
FC	0.000%	0.000%	0.000%
WSWL	42.857%	10.714%	0.000%
LIN-ET	71.429%	21.429%	0.000%
EXP-ET	60.714%	17.857%	3.571%
ACL	17.857%	3.571%	0.000%
LINET_A	42.857%	0.000%	7.143%
EXPET_A	50.000%	0.000%	0.000%
ACL_A	10.714%	0.000%	0.000%
LIN+EDD	46.429%	0.000%	0.000%
EXP+EDD	39.286%	0.000%	0.000%
ACL+EDD	14.286%	0.000%	3.571%
LIN+FC	35.714%	0.000%	7.143%
EXP+FC	39.286%	0.000%	0.000%
ACL+FC	14.286%	0.000%	0.000%
LIN+WSWL	53.571%	7.143%	0.000%
EXP+WSWL	35.714%	0.000%	0.000%
ACL+WSWL	10.714%	0.000%	0.000%
LIN+EDD_A	71.429%	3.571%	10.714%
EXP+EDD_A	57.143%	0.000%	3.571%
ACL+EDD_A	17.857%	0.000%	0.000%
LIN+FC_A	57.143%	3.571%	3.571%
EXP+FC_A	42.857%	0.000%	0.000%
ACL+FC_A	10.714%	0.000%	3.571%
LIN+WSWL_A	60.714%	0.000%	14.286%
EXP+WSWL_A	46.429%	3.571%	0.000%
ACL+WSWL_A	10.714%	0.000%	0.000%
SIM. ANNEL	82.143%	89.286%	39.286%
LOC.LIN-ET	82.143%	64.286%	3.571%

Tabela A2 - Valores para a medida P_h^* para as regras heurísticas implementadas

TAU	RHO	INSTANCIAS	EDD	FC	WSWL	LINET	EXPET	ACL	LINET_A	EXPET_A	ACL_A
0.2	0.2	1 E 15	14.864	13.421	5.425	0.469	0.461	0.690	0.956	0.978	0.891
0.2	0.4	2 E 16	14.621	9.331	9.453	1.798	2.053	1.934	1.642	1.581	1.890
0.2	0.6	3 E 17	14.777	9.728	12.993	3.303	3.304	1.774	1.623	1.537	2.230
0.2	0.8	4 E 18	14.974	10.056	16.538	2.141	3.227	0.746	2.387	1.771	0.879
0.2	1	5 E 19	14.339	9.882	19.019	0.980	1.164	1.043	3.470	2.587	1.588
0.4	0.2	6 E 20	18.491	22.676	10.166	5.645	8.093	5.392	1.744	2.293	4.246
0.4	0.4	7 E 21	18.781	23.480	21.012	5.974	6.175	5.170	0.751	0.705	2.818
0.4	0.6	8 E 22	19.580	25.926	32.379	10.181	11.055	8.165	1.856	5.325	7.351
0.4	0.8	9 E 23	20.720	19.888	38.571	9.762	13.713	6.653	7.344	7.476	4.673
0.4	1	10 E 24	17.021	16.549	39.320	9.416	10.021	8.038	6.116	6.951	5.344
0.6	0.2	11 E 25	26.356	33.488	15.409	7.108	6.602	10.988	2.160	3.016	9.185
0.6	0.4	12 E 26	28.165	36.993	25.894	10.574	11.437	8.156	3.430	3.064	6.903
0.6	0.6	13 E 27	28.957	42.957	33.296	8.776	8.780	8.104	2.190	5.481	7.595
0.8	0.2	14 E 28	26.245	34.882	9.099	3.698	3.698	1.851	0.642	3.865	2.231

Tabela A3 - Er_{min} segundo os factores de atraso e de dispersão das due dates

TAU	RHO	INSTANCIAS	LIN+EDD	EXP+ED	ACL+ED	LIN+FC	EXP+FC	ACL+FC	LIN+WSW	EXP+WSW	ACL+WS
0.2	0.2	1 E 15	1.860	2.269	2.270	0.598	3.006	1.966	1.949	2.429	2.027
0.2	0.4	2 E 16	1.007	1.360	1.917	1.599	1.923	2.994	1.336	2.719	1.975
0.2	0.6	3 E 17	1.798	1.311	1.159	1.898	1.539	1.589	2.367	1.289	1.930
0.2	0.8	4 E 18	2.496	1.906	0.935	2.482	1.690	0.993	2.330	1.718	1.709
0.2	1	5 E 19	2.369	2.252	1.244	3.240	2.259	1.312	2.889	2.268	1.362
0.4	0.2	6 E 20	0.911	2.456	4.083	1.578	2.213	4.614	1.075	1.856	4.748
0.4	0.4	7 E 21	0.921	3.071	2.851	1.495	2.720	3.483	1.780	3.388	4.162
0.4	0.6	8 E 22	3.033	4.279	5.822	7.357	7.907	6.864	4.833	5.351	8.428
0.4	0.8	9 E 23	4.755	3.530	3.762	5.210	4.516	3.942	4.680	5.053	5.050
0.4	1	10 E 24	5.975	6.796	6.492	6.714	7.201	5.411	6.166	6.269	5.976
0.6	0.2	11 E 25	2.704	5.864	10.465	2.413	5.768	10.437	1.844	4.684	9.348
0.6	0.4	12 E 26	3.137	3.099	5.955	3.831	3.830	5.424	4.284	4.772	6.627
0.6	0.6	13 E 27	4.535	5.178	8.560	3.805	4.791	8.331	2.126	5.602	8.182
0.8	0.2	14 E 28	1.440	4.242	3.541	0.825	4.146	3.317	0.514	3.815	3.306

Tabela A4 - Er_{min} segundo os factores de atraso e de dispersão das due dates

TAU	RHO	INSTANCIAS	LIN+EDD_A	EXP+EDD_A	ACL+EDD_A
0.2	0.2	1 E 15	5.239	3.471	3.025
0.2	0.4	2 E 16	6.050	3.634	6.406
0.2	0.6	3 E 17	4.398	4.361	3.246
0.2	0.8	4 E 18	4.106	4.071	3.475
0.2	1	5 E 19	4.906	5.060	3.579
0.4	0.2	6 E 20	5.628	5.223	6.743
0.4	0.4	7 E 21	5.252	11.004	7.991
0.4	0.6	8 E 22	11.151	4.958	8.139
0.4	0.8	9 E 23	9.574	11.532	8.046
0.4	1	10 E 24	8.405	9.291	10.905
0.6	0.2	11 E 25	11.005	11.163	14.390
0.6	0.4	12 E 26	14.604	15.731	18.569
0.6	0.6	13 E 27	12.410	14.804	17.853
0.8	0.2	14 E 28	9.033	11.653	10.563

Tabela A5 - Er_{min} segundo os factores de atraso e de dispersão das due dates

TAU	RHO	INSTANCIAS	LIN+FC_A	EXP+FC_A	ACL+FC_A
0.2	0.2	1 E 15	2.575	3.490	3.258
0.2	0.4	2 E 16	4.905	4.194	4.494
0.2	0.6	3 E 17	3.211	3.007	3.183
0.2	0.8	4 E 18	3.701	3.485	3.459
0.2	1	5 E 19	3.281	3.252	3.863
0.4	0.2	6 E 20	5.286	6.034	8.613
0.4	0.4	7 E 21	4.591	4.840	8.326
0.4	0.6	8 E 22	9.895	5.831	9.004
0.4	0.8	9 E 23	8.738	10.464	9.935
0.4	1	10 E 24	8.730	8.328	10.148
0.6	0.2	11 E 25	7.805	9.060	12.732
0.6	0.4	12 E 26	9.147	12.656	14.873
0.6	0.6	13 E 27	12.372	12.283	16.321
0.8	0.2	14 E 28	8.784	10.599	10.154

Tabela A6 - Er_{min} segundo os factores de atraso e de dispersão das due dates

TAU	RHO	INSTANCIAS	LIN+WSWL_A	EXP+WSWL_A	ACL+WSWL_A
0.2	0.2	1 E 15	3.775	3.781	6.140
0.2	0.4	2 E 16	4.070	3.733	5.031
0.2	0.6	3 E 17	3.625	3.303	4.256
0.2	0.8	4 E 18	3.137	3.971	3.580
0.2	1	5 E 19	4.104	3.979	4.906
0.4	0.2	6 E 20	5.408	5.186	6.081
0.4	0.4	7 E 21	5.101	9.551	7.913
0.4	0.6	8 E 22	10.029	6.374	12.283
0.4	0.8	9 E 23	9.658	11.124	9.853
0.4	1	10 E 24	8.879	10.455	10.642
0.6	0.2	11 E 25	7.968	11.223	14.480
0.6	0.4	12 E 26	13.528	15.151	16.439
0.6	0.6	13 E 27	13.179	12.946	19.773
0.8	0.2	14 E 28	8.894	10.048	10.196

Tabela A7 - Er_{min} segundo os factores de atraso e de dispersão das due dates

TAU	RHO	INSTANCIAS	ANNEALING	PES.LOC
0.2	0.2	1 E 15	0.230	0.461
0.2	0.4	2 E 16	1.074	1.794
0.2	0.6	3 E 17	0.000	2.466
0.2	0.8	4 E 18	0.000	0.752
0.2	1	5 E 19	0.000	0.292
0.4	0.2	6 E 20	0.992	5.504
0.4	0.4	7 E 21	0.265	5.929
0.4	0.6	8 E 22	4.179	8.514
0.4	0.8	9 E 23	0.958	9.368
0.4	1	10 E 24	0.000	8.976
0.6	0.2	11 E 25	1.537	7.093
0.6	0.4	12 E 26	4.034	10.574
0.6	0.6	13 E 27	0.889	8.691
0.8	0.2	14 E 28	0.132	3.692

Tabela A8 - Er_{min} segundo os factores de atraso e de dispersão das due dates

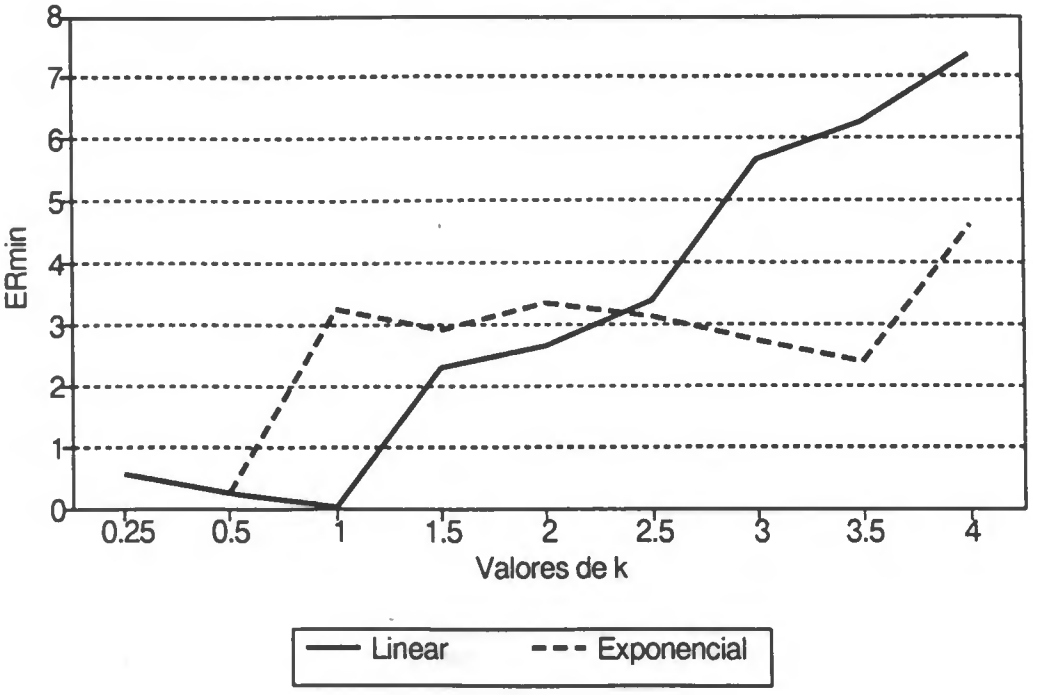


Gráfico A1

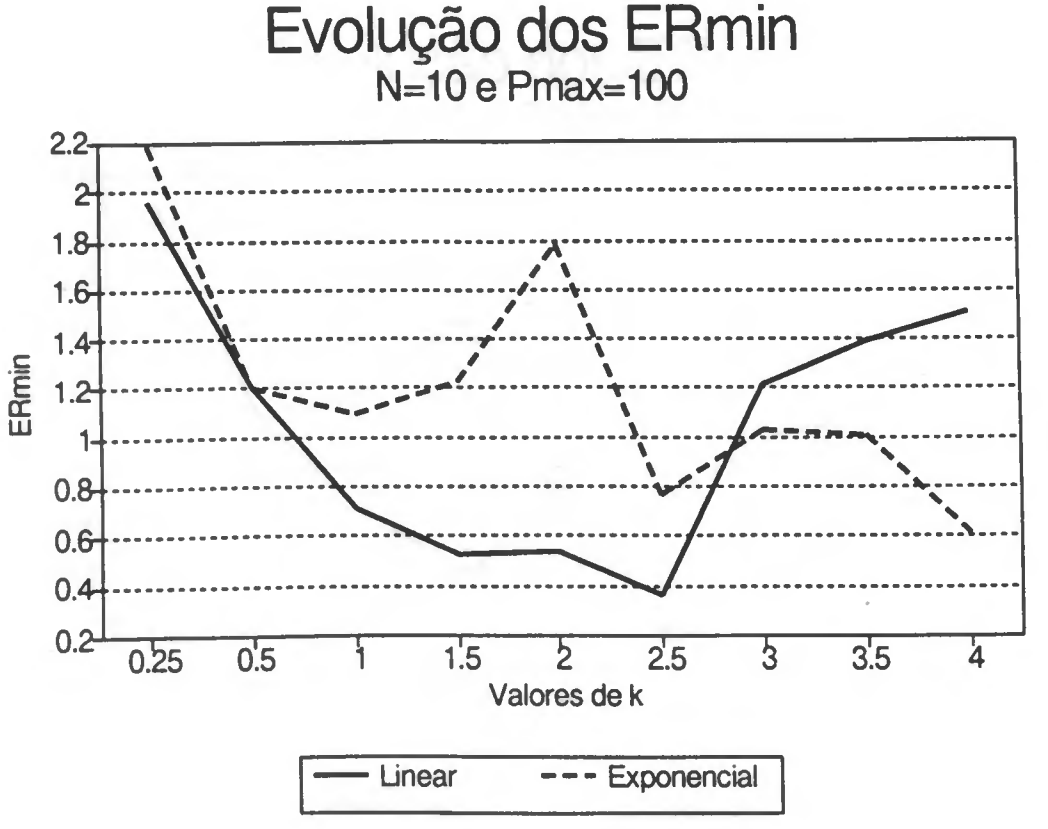


Gráfico A2

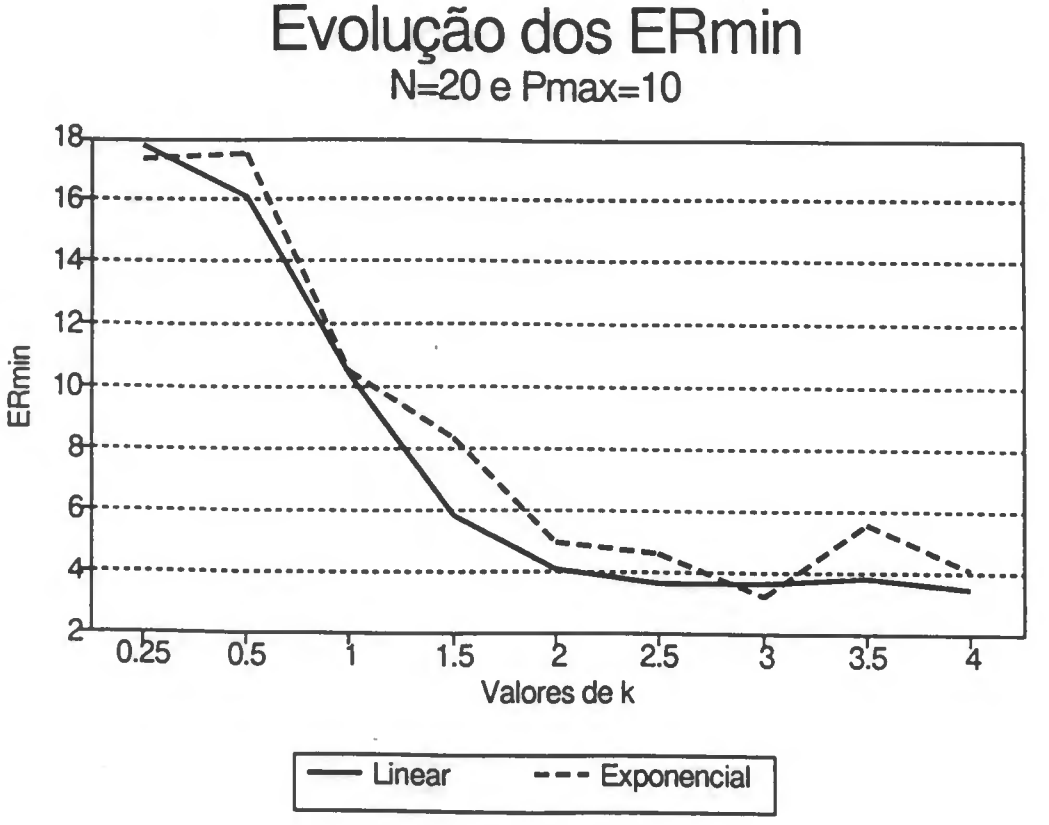


Gráfico A3

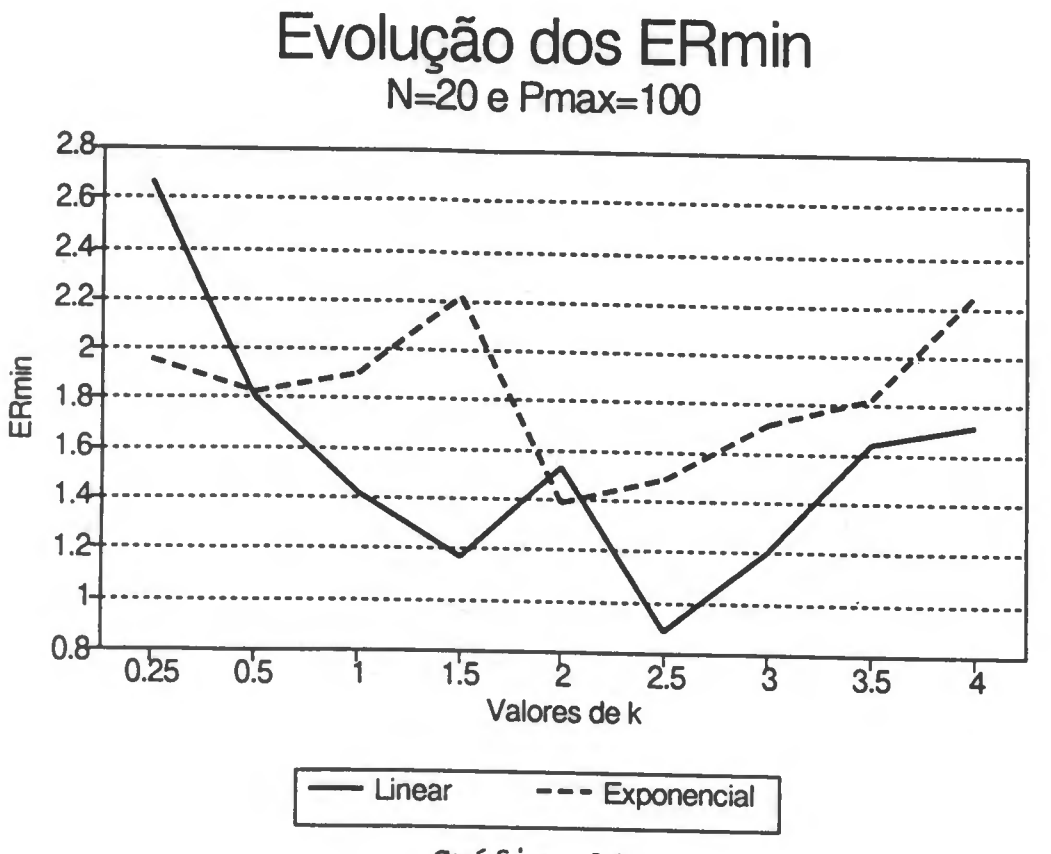


Gráfico A4

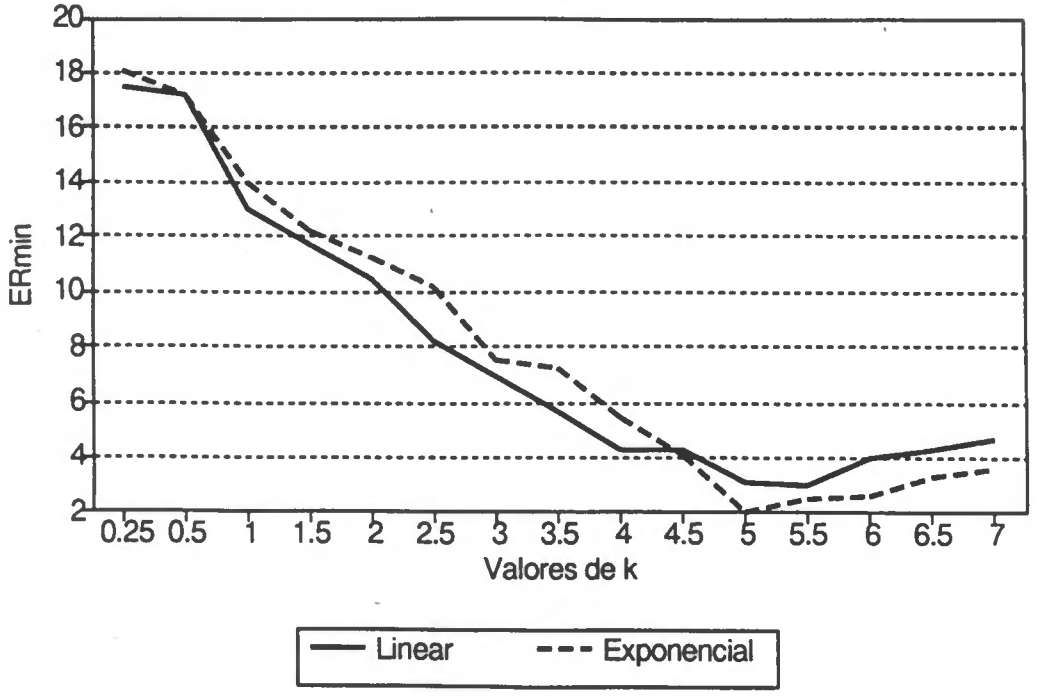


Gráfico A5

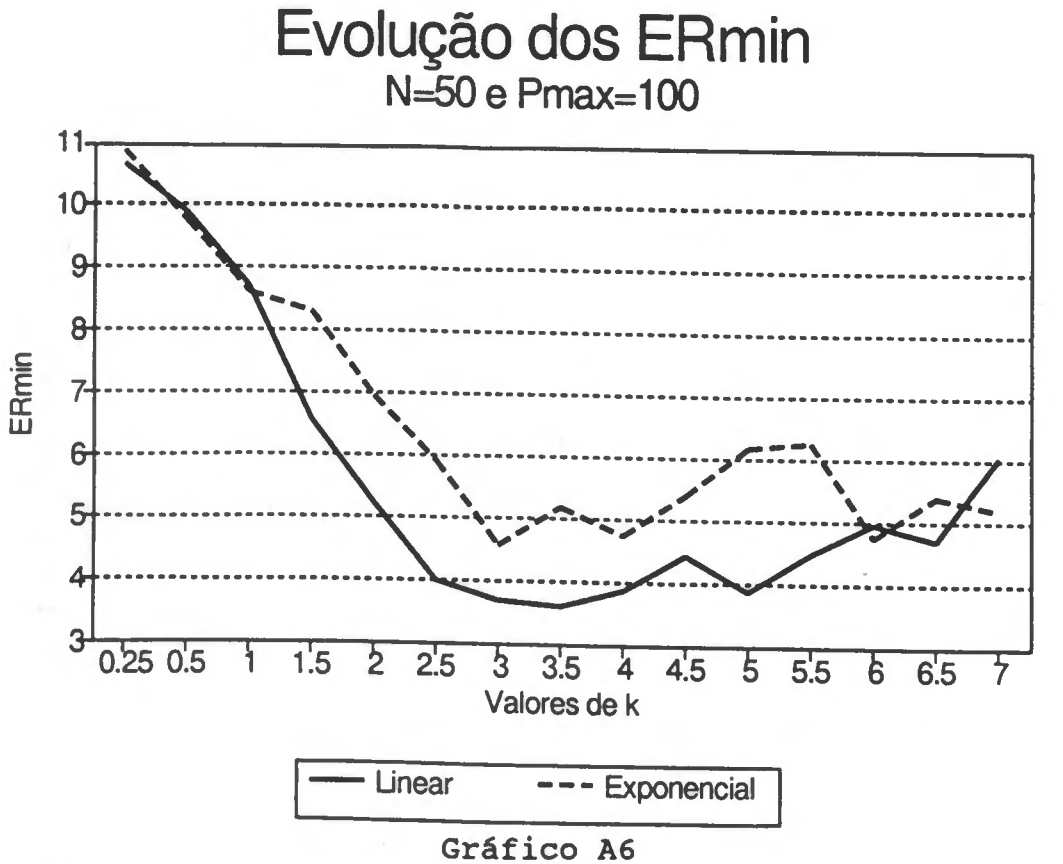


Gráfico A6